Janne Mäyrä

Kvanttitietokoneen toiminnan simulointi ja emulointi

Tietotekniikan kandidaatintutkielma

30. lokakuuta 2017

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Janne Mäyrä

Yhteystiedot: jaeeolma@student.jyu.fi

Ohjaaja: Sanna Mönkölä

Työn nimi: Kvanttitietokoneen toiminnan simulointi ja emulointi

Title in English: Simulation and emulation of a quantum computer

Työ: Kandidaatintutkielma

Sivumäärä: 24+0

Tiivistelmä: Tämän tutkielman tavoitteena on antaa lukijalle yleiskuva kvanttitietokoneiden toiminnasta, simuloinnin ja emuloinnin eroista sekä konkreettiset esimerkit kvanttitietokoneen toiminnan simuloinnista ja emuloinnista. Lisäksi tarkoitus on selventää syitä sille, miksi kvanttitietokoneiden toimintaa pyritään mallintamaan niiden rakentamisen sijaan. Johtopäätöksenä todetaan, että vaikka nykyisillä simulointi- ja emulointimenetelmillä ei merkittävää suorituskyvyn parannusta klassisiin tietokoneisiin nähden saada, mallintamalla saadaan tärkeää tutkimustietoa tehokkaampien mallinnusmenetelmien kehittämisestä.

Avainsanat: kvanttitietokone, simulointi, emulointi

Abstract: This paper is meant to introduce reader to quantum computing, differences between simulation and emulation and to give examples on how to both simulate and emulate a quantum computer. In addition, it is meant to clarify why instead of building a quantum computer one would rather simulate or emulate it. Even though simulation or emulation doesn't give significant increases to computation power, modeling quantum computers this way gives important new information to development of new methrods of simulation and emulation.

Keywords: quantum computing, simulation, emulation

Kuviot

Sisältö

1 Johdanto

Kandidaattitutkielmassa toteutettiin kirjallisuuskatsaus kvanttitietokoneen toiminnan simulointi- ja emulointimenetelmiin. Tavoitteena on antaa yleiskuva siitä, miten ja miksi kvanttitietokoneita simuloidaan ja emuloidaan, mitä eroa näillä on, sekä mitä haasteita tai rajoitteita näissä on.

Konsepti kvanttitietokoneista esiteltiin jo 1980-luvulla useiden tutkijoiden toimesta (Deutsch, Benioff, Feynman), ja 1990-luvun alussa onnistuttiin näyttämään, että ne ovat huomattavasti tehokkaampia kuin klassiset tietokoneet tiettyjen tehtävien suorittamisessa. Tunnetuimpia tällaisia algoritmeja ovat Peter Shorin vuonna 1994 esittelemä *Shorin algoritmi*, jonka avulla voidaan etsiä minkä tahansa kokonaisluvun alkulukukertoimet polynomiaalisessa ajassa (aikavaativuus $O(logN)$), kun tehokkaimmat vastaavat klassisilla tietokoneilla toimivien algorimien aikavaativuus on lähes eksponentiaalinen (Shor (1999)), sekä Lov Groverin vuonna 1996 esittelemä *Groverin algoritmi*, jonka avulla N alkioisesta tietokannasta löydetään haluttu alkio nopeimmillaan $O(\sqrt{N})$ ajassa (klassisilla tietokoneilla tähän kuluu vähintään $O(N)$ operaatiota) (Grover (1996)).

Varsinaisen kvanttitietokoneen rakentamisessa on useita haasteita. Tämänhetkiset toteutukset perustuvat esimerkiksi suprajohteisiin tai fotonien kaappaamiseen. Tällöin järjestelmän tila täytyy pystyä pitämään mahdollisimman häiriöttömänä. Pienikin häiriö kvanttijärjestelmän koherenssiin pudottaa suoritustehoa erittäin paljon. Tästä syystä kvanttitietokoneen toimintaa mallinnetaan joko simuloimalla tai emuloimalla klassisessa tietokoneympäristössä (La Cour, Ostrove, Ott, Starkey & Wilson (2016)). Mallintamisen tavoitteena on paitsi saada lisätietoa kvanttitietokoneiden toiminnasta ennen konkreettisten laitteiden rakentamista, myös varmistua algoritmien oikeelllisuudesta testaamalla ja debuggaamalla niitä mallinnusympäristöissä (Häner, Steiger, Smelyanskiy & Troyer (2016)).

Tällä hetkellä markkinoilla olevista kvanttitietokoneista esiin voidaan nostaa D-Wave Systemsin laitteet sekä IBM:n Quantum Experience (QX). D-Wave One on

maailman ensimmäinen myynnissä oleva kvanttitietokoneena markkinoitu laite, jossa on valmistajan ilmoituksen mukaan 108 kubitin rekisteri. Uudemmassa D-Wave Two -laitteessa rekisterin koko on kasvanut yli viiteensataan kubittiin, ja D-Wave 2000Q on jo kahden tuhannen kubitin tietokone. D-Waven koneet eivät kuitenkaan ole yleiskäyttöisiä, vaan niillä voidaan ratkaista vain tietyllä tavalla muotoiltuja optimointitehtäviä. Tästä johtuen ne eivät kaikkien mielestä ole ”aitoja” kvanttitietokoneita, vaikka ne hyödyntävätkin kvanttimekaniikan ilmiöitä (Shin, Smith, Smolin & Vazirani (2014)). Lisäksi eräässä tutkimuksessa todettiin, että D-Wave Two -koneella ei välttämättä saada ratkaistua ongelmia nopeammin kuin klassisilla tietokoneilla (Rønnow, Wang, Job, Boixo, Isakov, Wecker, Martinis, Lidar & Troyer (2014)). IBM QX on kenen tahansa vapaasti ohjelmoitavissa oleva, tällä hetkellä viiden kubitin rekisterillä oleva kvanttitietokone. IBM QX on valmistajansa mukaan yleiskäyttöinen kvanttitietokone, toisin kuin D-Waven laitteet, mutta viiden kubitin rekisterillä ei ole vielä mahdollista ratkaista vaativia ongelmia. IBM QX:n seuraavassa versiossa on 16 kubitin rekisteri, ja se on tutkielman kirjoitushetkellä betatestauksessa (IBM (2017)).

Termeinä simulointi ja emulointi vaikuttavat samanlaisilta, ja arkikielessä niitä toisinaan käytetään toistensa synonyymeina. Molemmilla termeillä tarkoitetaan jonkin asian toteuttamista vieraassa ympäristössä, mutta hieman eri tavoilla. *Simuloitaessa* pyritään toteuttamaan simuloitavan kohteen *käytöstä*, kun taas *emuloitaessa* pyritään toteuttamaan emuloitavan kohteen *sisäinen toiminta*. Konkreettisena esimerkkinä tästä voidaan käyttää esimerkiksi videopelejä. Lentosimulaattori simuloi lentokoneen ohjaamista, kun taas emulaattorin avulla voidaan emuloida toisen pelikonsolin järjestelmän toimintaa, ja näin pelata kyseiselle konsolille tehtyjä pelejä täysin toisessa ympäristössä.

Huolimatta siitä, halutaanko kohteen toimintaa simuloida vai emuloida, on tietyt asiat otettava huomioon tapauksesta riippumatta. Sekä simuloidun että emuloidun mallin on oltava riittävän totuudenmukaisia ja tarkkoja, jotta saadut tulokset ovat tilastollisesti merkittäviä. Se, miten tarkkoja tuloksia vaaditaan, riippuu projektin tavoitteista. Käytännössä mitä pienempää kohdetta mallinnetaan, sitä tarkemmin on

jokainen yksittäinen asia on otettava huomioon. Lisäksi sekä simulointi että emulointi on suositeltavaa toteuttaa pelkistämättä komponentteja, eli toimia kuten reaalimaailman vastineensa(McGregor (2002)).

Simuloinnin ja emuloinnin välillä on samankaltaisista käyttötarkoituksista huolimatta useita eroavaisuuksia sekä tavoittteiden että konkreettisen toiminnan osalta. Simulaatiomalleilla testataan yleensä useita eri ratkaisuja ja etsitään parasta mahdollista vaihtoehtoa, kun taas emulaatiomalleilla tällaisia testejä ei vastaavalla tavalla voi edes suorittaa. Koska emulaatiomallissa mallinnetaan kokonaisen järjestelmän sisäistä toimintaa, ei sen avulla voida vastaavalla tavalla keskittyä yksittäisen toiminnon tutkimiseen. Emulaatiomallin toiminnassa kaikki päätökset ja toiminnot tapahtuvat reaaliajassa, koska myös niiden mallintamat järjestemät toimivat reaaliajassa. Simulaatiomallit puolestaan suunnitellaan mahdollisimman nopeiksi. McGregor (2002) antaa esimerkkitapaukseksi liukuhihnojen risteykseen saapuvaa laatikkoa. Laatikon suunta risteyksestä riippuu sisällöstä ja lopullisesta määränpäästä. Simulaatiossa tämä päätös tapahtuu välittömästi, reaaliajassa tähän voi kuitenkin kuulua useita vaiheita (McGregor (2002)).

Onnistuneen simulaatiomallin tärkeimpänä piirteenä McGregor pitää toistettavuutta. Simuloitaessa tulee saada aina samoilla parametreillä identtiset tulokset, ja kaikki satunnaisuus syntyy esimerkiksi rajoitetun satunnaislukugeneraattorin vaikutuksesta. Emulaatiomallin on taas oltava robusti(vakaa), sillä mallissa on otettava huomioon muun muassa mallin ja emuloivan järjestelmän kellojen välinen ero ja kommunikoinnissa tuleva viive (McGregor (2002)).

Tutkielman rakenne on seuraava: Toisessa luvussa esitellään yleisesti kvanttitietokoneiden toimintaperiaate ja keskeisimmät käsitteet siihen liittyen. Kolmannessa luvussa perehdytään kvanttitietokoneen simulointiin yleisesti, ja esitellään sekä supertietokoneella että näytönohjaimella tehtävää simulaatiota. Neljännessä luvussa perehdytään vastaavalla tavalla emulointiin, ja viidennessä luvussa pohditaan työstä nousseita johtopäätöksiä.

2 Kvanttitietokoneen toiminnasta

Eräs merkittävä, ellei merkittävin, ero kvanttitietokoneiden ja klassisten tietokoneiden välillä on näiden tapa tallentaa tietoa. Klassisten tietokoneiden käyttämillä biteillä on joko arvo 0 tai arvo 1, kun taas kvanttibitillä eli kubitillä voidaan tulkita olevan arvot 0 ja 1 yhtäaikaisesti, eli kubitti on niin kutsutussa superpositiossa. Tällöin algoritmit ja porttioperaatiot voivat käsitellä useita arvoja yhdessä vaiheessa verrattuna klassisiin tietokoneisiin, jotka voivat käsitellä vain yhtä arvoa kerrallaan. Kuitenkin siinä vaiheessa, kun kubittia halutaan tarkastella, se romahtaa joko arvoon 0 tai 1, eikä siis ole enää superpositiossa. Sille, kumpaan arvoon kubitti romahtaa, on olemassa tietty todennäköisyys (Esim. Nielsen & Chuang (2002), Elhoushi, El-Kharashi & Elrefaei (2011)).

Toinen merkittävä ero näiden kahden tietokoneen välillä on yleinen toimintatapa. Kun klassiset tietokoneet toimivat deterministisesti, eli laskennan tulokset saadaan mitattua ilman virheitä, ovat kvanttitietokoneet probabilistisiä eli todennäköisyyksiin perustuvia. Tästä johtuen kvanttitietokoneella tehtävät laskelman joudutaan tekemään useita kertoja mittausvirheen pienentämiseksi (Khalid, Zilic & Radecka (2004)).

Tarkemmin määriteltynä kubitin tila ja superpositio voidaan määritellä seuraavasti: kubitin perustilat ovat siis $|0\rangle$ ja $|1\rangle$, mutta kubitti voi siis myös olla tilojen lineaarikombinaatiossa eli superpositiossa, joka määritellään seuraavasti:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

missä α ja β ovat kompleksilukuja, joskin Nielsen & Chuang (2002) mukaan ne voidaan useimmiten ajatella reaalilukuina ilman käytännön ongelmia. Käytännössä tämä tarkoittaa sitä, että kubitin tila voidaan määritellä vektorina kaksiuloitteisessa kompleksisessa vektoriavaruudessa, jonka kantavektoreita ovat $|0\rangle$ ja $|1\rangle$.

Silloin, kun kubittia mitataan, se siis romahtaa joko tilaan $|0\rangle$ tai $|1\rangle$, ja kuten aiemmin on todettu, molemmille on tietty todennäköisyys. Nämä todennäköisyydet tilalle $|0\rangle$ $|\alpha|^2$ ja tilalle $|1\rangle$ $|\beta|^2$, eli luonnollisesti $|\alpha|^2 + |\beta|^2 = 1$. Esimerkiksi superpo-

sitio, joissa molemmilla tiloilla on sama todennäköisyys, voidaan ilmoittaa seuraavasti:

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

joka voidaan merkitä myös $|+\rangle$.

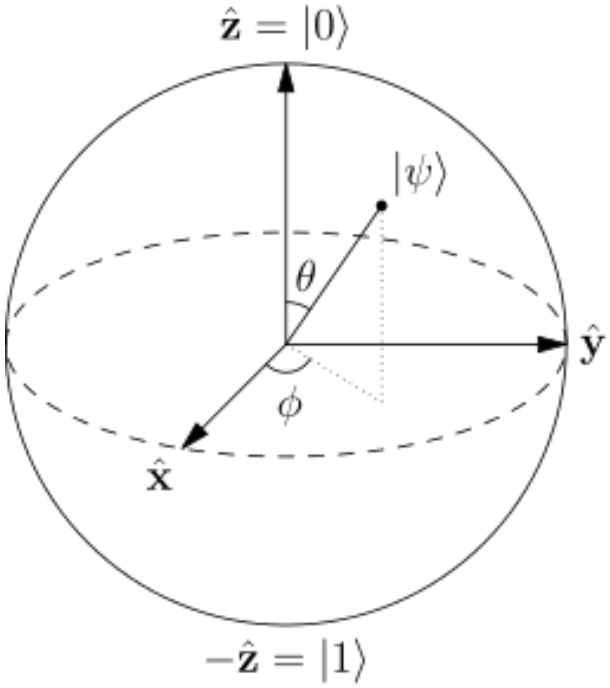
Kun kubittia tarkastellaan, saadaan tulokseksi aina joko $|0\rangle$ tai $|1\rangle$, ja tällöin mitattavan kubitin tila muuttuu pysyvästi täksi tilaksi. Esimerkiksi, vaikka $|+\rangle$ on mitattaessa samalla todennäköisyydellä kumpi tila tahansa, jos sen mittaustulokseksi saadaan $|1\rangle$, mittauksen jälkeinen tila on $|1\rangle$. Entä jos kubittia ei mitattaisikaan ollenkaan? Paljonko tietoa silloin pystytään ilmaisemaan yhdellä kubitilla? Tätä potentiaalista, piilotettua tietoa kutsutaan *kvantti-informaatioksi*.

Jos kubitteja on useampi kuin yksi, kvantti-informaation määrä kasvaa eksponentiaalisesti. Kahden tavallisen bitin mahdolliset tilat ovat 00, 01, 10 ja 11, niin aivan vastaavasti kahden kubitin mittauksenjälkeiset perustilat ovat $|00\rangle$, $|01\rangle$, $|10\rangle$ ja $|11\rangle$. Koska ennen mittausta nämäkin kaksi kubittia voivat olla näiden tilojen välisessä superpositiossa, käytetään todennäköisyysvektorin laskemisessa kompleksista kerrointa, jota usein kutsutaan amplitudiksi. N-kubittisella järjestelmällä on aina 2^n amplitudia, joten esimerkiksi 500-kubittisen järjestelmän amplitudien laskeminen on klassisille tietokoneille käytännössä mahdotonta (Nielsen & Chuang (2002)).

Varsinainen kvanttitietokone koostuu toisiinsa kytketyistä *kvanttiporteista*, aivan kuten klassinen tietokone koostuu toisiinsa kytketyistä loogisista porteista. Teoriassa kvanttiportteja on äärettömän paljon, mutta Nielsenin mukaan kolme yleisimmin käytettyä ja samalla tunnetuinta yhden kubitin kvanttiporttia ovat NOT-portti (useimmiten merkitään symbolilla X), Z-portti ja *Hadamard*-portti (merkitään yleensä symbolilla H). Esimerkiksi X-portti muuttaa yhtälön $\alpha|0\rangle + \beta|1\rangle$ muotoon $\alpha|1\rangle + \beta|0\rangle$, Z-portti ei tee mitään muutoksia $|0\rangle$-tilalle, mutta muuttaa $|1\rangle$ muotoon $-|1\rangle$. H-portti on Nielsenin mukaan eräs hyödyllisimmistä kvanttiporteista, sillä sen muokkaaman kubitin todennäköisyydet päätyä eri tilaan ovat yhtä suuret (Nielsen &

Chuang (2002)). Muita portteja ovat esimerkiksi Y-portti ja *Phase shift*-portti (Gutiérrez, Romero, Trenas & Zapata (2010)).

Eräs tapa havainnollistaa yksittäistä kubittia ja sille tehtäviä porttioperaatioita on *Blochin pallo*, joka on kolmiulotteisessa avaruudessa sijaitseva pallo (kuvio 1). Pallon pohjoisnapa vastaa tilaa $|0\rangle$ ja etelänapa tilaa $|1\rangle$, ja ennen mittausta kubitin voidaan ajatella sijaitsevan jossain kohtaa pallon kehällä riippuen parametrien α ja β arvoista. Myös porttioperaatioita voi havainnollistaa Blochin pallon avulla. Esimerkiksi X-portti pyörittää palloa π radiaanin verran x-akselin ympäri, Z-portti vastaavasti π radiaania z-akselin ympäri ja Y-portti y-akselin ympäri, H-portti puolestaan pyörittää palloa ensin y-akselin ympäri $\pi/2$ radiaanin verran, ja sen jälkeen peilaa vektorin xy-tason suhteen. Blochin palloa vastaavaa yksinkertaista esitystapaa ei kuitenkaan ole löydetty kahden tai useamman kubitin muodostamalle järjestelmälle. (Nielsen & Chuang (2002), Shor (1999)).



Kuvio 1. Blochin pallo

Kahden kubitin porteista merkittävin on CNOT-portti(*Controlled NOT*), sillä sen ja yhden kubitin porttien avulla voidaan muodostaa yleiset porttioperaatiot n-kubit-

tisille järjestelmille. Lisäksi, Gottesman-Knillin teoreeman mukaan mikä tahansa pelkästään CNOT- ja Hadamard -porteista koostuva kvanttipiiri voidaan simuloida tehokkaasti klassisella tietokoneella (Gottesman (1998)). CNOT-portille annetaan syötteenä kaksi kubittia, joista ensimmäistä kutsutaan kontrollibitiksi ja toista kohdebitiksi. Mikäli kontrollibitin tila on $|0\rangle$, ei kohdebitille tehdä muutoksia, mutta mikäli kontrollibitti on tilassa $|1\rangle$, kohdebitin tila vaihtuu päinvastaiseksi. Mahdolliset operaatiot ovat siis $|00\rangle \rightarrow |00\rangle$, $|01\rangle \rightarrow |01\rangle$, $|10\rangle \rightarrow |11\rangle$ ja $|11\rangle \rightarrow |10\rangle$. Hieman samalla periaatteella toimii merkittävin kolmen kubitin portti, *Toffoli-portti*. Toffoli-portille annetaan syötteenä kolme kubittia, joista kaksi ensimmäistä ovat kontrollibittejä. Mikäli molemmat kontrollibitit ovat tilassa $|1\rangle$, suoritetaan kohdebitille jokin porttioperaatio, esimerkiksi X (Juliá-Díaz, Burdis & Tabakin (2006)).

Koska kubitit romahtavat pysyvästi tiettyyn tilaan mittausvaiheessa, on niitä pystyttävä manipuloimaan ennen mittausta. Kvanttimekaniikan lait sallivat kuitenkin pelkästään *unitaarisia* muunnosoperaatioita tilavektoreille. Matriisille U pätee, että se on unitaarinen, jos sen konjugaattitranspoosi on sama kuin sen käänteismatriisi, eli pätee, että $U^\dagger U = I$. Tämä pätee luonnollisesti myös aiemmin mainittuja portteja vastaaville matriiseille. Huomioitavaa on, että koska unitaarisuus on ainoa vaatimus kvanttiportille, on niitä teoriassa äärettömän monta. Käytännössä portteja ei kuitenkaan tarvita kuin muutama (Shor (1999), Nielsen & Chuang (2002)).

Kvanttilaskennalle voidaan asettaa Nielsenin mukaan neljä perusvaatimusta: kvantti-informaatio on pystyttävä esittämään vakaasti, unitaarimuunnoksia on pystyttävä suorittamaan riittävä määrä, kubitit on pystyttävä alustamaan selkeään alkutilaan ja lopputila on pystyttävä mittaamaan luotettavasti. Nämä vaatimukset ovatkin pääosin myös kvanttialgoritmien suoritusvaiheet: ensin kubitit alustetaan alkutilaan (käytännössä aina $|0\rangle$), sen jälkeen ne ajetaan porttitaulukon läpi ja lopulta niiden arvot mitataan (Nielsen & Chuang (2002)).

3 Simulointi

Simuloinnin vahvuus verrattuna jopa konkreettiseen kvanttitietokoneeseen on toisaalta se, että simulaatiossa saadaan aina haluttu tulos, eivätkä tähän tulokseen vaikuta mitkään ulkopuoliset häiriötekijät. Esimerkiksi IBM QX -kvanttitietoko- neella yhdelle kubitille tehdyssä X-operaatiossa päädytään laskennallisesti virheelliseen tulokseen noin 7% ajasta, kun taas simuloidessa tulos on aina laskennallisesti oikea (kuvio 2).

Kubittien määrän kasvaessa kohdataan kuitenkin ongelmia. Suurimmat haasteet liittyvät rinnakkaisuuden simuloimiseen. Koska samanaikaisesti laskettavan informaation määrä kasvaa eksponentiaalisesti kubittien määrän kasvaessa, tehokkaan simulointitavan löytäminen on hyvin haastavaa, ellei jopa mahdotonta. Kuten aiemmin mainittua, simulaatio ei voi edetä pisteestä toiseen ennen kuin kaikki vaaditut laskutoimitukset on suoritettu. Tästä johtuen kvanttitietokoneen toimintaa simuloitaessa käytetään usein rinnakkaisia laitteita, kuten CUDA-rajapinnan tapauksessa tietokoneen prosessoria ja näytönohjainta (Gutiérrez ym. (2010)).

Konkreettinen esimerkki rinnakkain suoritettavien laskutoimitusten ja muistissa pidettävien tilojen määrästä on esimerkiksi seuraava: Shorin algoritmin yhden askeleen mallintamista varten vaaditaan $3^{16} = 43046721$ erillisen tilan tallentamista, kun $N = 15$. Lisäksi tässä simulaatiossa on yli 8000 erillistä vaihetta, joista jokainen tulee suorittaa jokaiselle tilalle (Obenland & Despain (1998)).

Aiemmin mainitun Gottesman-Knillin teoreeman mukaan kvanttipiiriä, joka koostuu pelkästään tilojen alustamisesta, niin kutsutusta Clifford-porttiperheestä (CNOT, X, Y, Z, Hadamard ja Phase) ja lopputilojen mittaamisesta, voidaan simuloida tehokkaasti klassisella probabilistisella tietokoneella (Gottesman (1998), Nielsen & Chuang (2002)). Tämä teesi ei kuitenkaan tarkoita, että kvanttitietokoneita olisi turha rakentaa. Teoreema pikemminkin antaa ne rajoitteet, joiden mukaisia piirejä voidaan simuloida. Tällä porttiperheellä ei kuitenkaan hyödynnetä kvanttilomittumista. Klassisella tietokoneella tarkoitetaan järjestelmää, joka noudattaa tunnettuja fy-

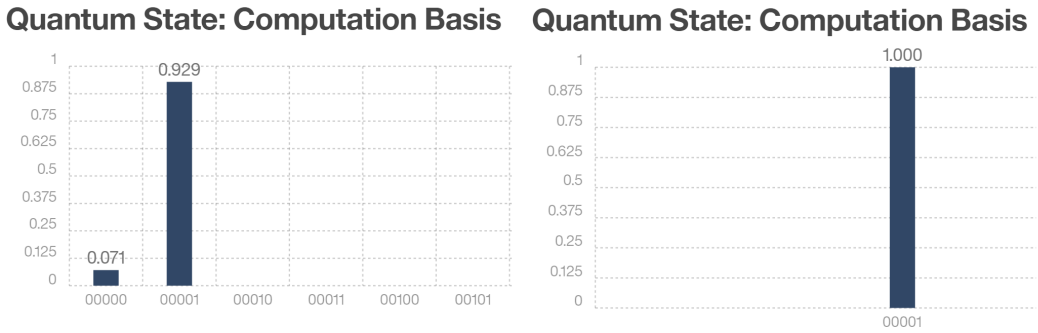
siikan lakeja ja kaikki sen tekemät toimenpiteet voidaan esittää klassisesti. Kysymys on pikemminkin siitä, onko mahdollista rakentaa tietokone, joka laskee kvanttimekaanisia ilmiöitä vastaavia todennäköisyyksiä riittävällä tarkkuudella (Feynman (1982), Cuffaro (2015)).

Kvanttialgoritmien simulointiympäristöt noudattavat jotain virtuaalista laitteistomallia, yleisimmin joko kvanttipiirimallia tai QRAM-mallia. Kvanttipiirimalli muodostuu useista kvanttiporteista, jotka on kytketty toisiinsa vastaavasti kuin klassiset loogiset portit. QRAM-mallissa puolestaan joukolle kubitteja tai kvanttirekisterejä annetaan klassiselta tietokoneelta komentoja suoritettavaksi. Esimerkki tällaisesta komennosta on "mittaa kubitti q0". Kolmas yleisesti tunnettu malli, Turingin kvanttitietokone, on puhtaasti teoreettinen, joten sitä ei tässä esitellä sen tarkemmin. Sen voidaan kuitenkin osoittaa olevan toiminnaltaan kahta muuta mallia vastaava (Khalid ym. (2004)). Suorituskyvyllisesti sekä kvanttipiirimalli että QRAM-malli ovat, ainakin teoreettisesti, samanlaiset. Merkittävimmät erot niiden välillä ovat perusoperaatioiden painotuksissa. Kvanttipiirimallissa mittaukset suoritetaan aina laskennan viimeisenä vaiheena kaikille kubiteille, kun taas QRAM-mallissa mittauksia ja porttioperatioita voidaan suorittaa vapaammassa järjestyksessä. Selingerin (2004) mukaan tämän johdosta QRAM-malli sopii paremmin ainakin kvanttiohjelmointikielien vaatimuksille (Selinger (2004)).

Tässä luvussa perehdytään kahteen erilaiseen simulaatiomalliin: Useiden tietokoneiden rinnakkaislaskennan avulla toimiviin malleihin, kuten Harvardin yliopistossa kehitettyyn qHiPSTER-simulaattoriin (*the Quantum High Performance Software Testing Environment*) ja NVIDIAn kehittämän CUDA-rajapinnan (*The Compute Unified Device Architecture*) avulla suoritettavan simulointiin.

3.1 Kvanttitietokoneen simulointi CUDA-rajapinnalla

CUDA on NVIDIAn kehittämä rinnakkaislaskennan mahdollistava ohjelmointirajapinta, jonka avulla pystytään hyödyntämään näytönohjaimen laskentatehoa erilaisten laskutoimitusten suorittamisessa. Nykyaikaiset näytönohjaimet koostuvat



Kuvio 2. Yhdelle kubitille suoritetun X-operaation laskentatulokset IBM QX -kvanttitietokoneella (vasen) ja simulaatiossa (oikea)

useista rinnakkaisista suorittimista, joten käyttämällä niitä laskentaan pystytään esimerkiksi kvanttitietokoneen rinnakkaisuutta mallintamaan suhteellisen tehokkaasti, ja täten simulaatio saadaan vietyä läpi järkevässä ajassa verrattuna yhdellä suorittimella tehtävään simulaatioon. Gutiérrez, Romero, Trenas & Zapata (2010) esittävät artikkelissaan pohjimmiltaan QRAM-arkkitehtuurin mukaisen simulaatiomallin, jolla he simuloivat maksimissaan 26-kubittisen kvanttitietokoneen toimintaa. Mallissa kvanttitietokonetta vastaa näytönohjain, ja tietokoneen prosessori antaa komennot (Gutiérrez ym. (2010)).

Pohjimmiltaan näytönohjain koostuu taulukosta SIMT-multiprosessoreja (*Single Instruction Multiple Threads*), joista jokainen vuorostaan koostuu useista virtaprosessoreista (*Stream Processor*). CUDA-malli perustuu näytönohjaimen abstraktiotasojen hierarkiaan: hiloihin (*grids*), lohkoihin (*blocks*), warppeihin (*warps*) ja säikeisiin (*threads*). Näistä säikeet ovat yhteen prosessoriin sidottuja suoritusyksiköitä. Lohkot ovat yhteen multiprosessoriin sidottu joukko säikeitä, ja hila puolestaan koostuu useista lohkoista. Huomioitavaa on, että koska lohkoja voi olla useampia kuin multiprosessoreja, joudutaan lohkot vuorontamaan tietyille multiprosessoreille. Warp vuorostaan on joukko tietyn ohjeistuksen mukaisen suorituksen tekeviä säikeitä. Tällöin ohjelmaa suoritettaessa on mahdollista suorittaa useita laskutoimenpiteitä tehokkaasti rinnakkain (Gutiérrez ym. (2010)).

Laskennallisesti amplitudien joukkoa merkitään mallissa $S = \{\alpha_0, \alpha_1, ..., \alpha_{N-1}\}$, mis-

sä $N = n^2$. Sekä alkutilalle että lopputilalle on omat suljetut joukkonsa *in* ja *out*, jotka täyttävät tietyt mallille asetetut määritelmät. Mallin tarkoituksena on simuloida n-kubittisen rekisterin lopputilaa tietyn muunnosoperaation jälkeen, tässä tapauksessa joko Walsh-muunnoksen eli H-portin tai kvanttifouriermuunnoksen (*QFT*). QFT toteutettiin mallissa H-portin ja Controlled Phase Shift -portin yhteisoperaationa. Tällaisten laskutoimitusten aikavaativuus yleisesti on $O(2^{2n})$. Mallin ohjelmakoodi jakautuu CUDA-arkkitehtuurin mukaisesti prosessorissa suoritettavaan (isäntä, *host*) ja näytönohjaimessa suoritettavaan (ydin, *kernel*) koodiin. Ytimessä voidaan laskea useita amplitudeja rinnakkain, koska amplitudien joukot ovat suljettuja ja täten riippumattomia toisistaan (Gutiérrez ym. (2010)).

Simulaatiomallissa jokaista amplitudia vastaa 32-bittinen liukuluku, jolloin käytössä ollut laitteisto rajasi simulaation kapasiteetiksi 26 kubittia. Lopputuloksena Gutiérrez ym päätyivät arvioon, että tämänkaltaisella rinnakkaislaskentamallilla pystytään simuloimaan 26-kubittista rekisteriä jopa sata kertaa nopeammin kuin yksiprosessorisella järjestelmällä (Gutiérrez ym. (2010)).

CUDA ei ole ainoa rajapinta, jolla kvanttioperaatioita voidaan malllintaa prosessorin ja näytönohjaimen rinnakkkaislaskennan avulla. Muitakin vastaavia rajapintoja on olemassa, kuten esimerkiksi Khronos Groupin OpenCL. OpenCL on avoin ja yleiskäyttöisempi rajapinta verrattuna CUDAan, koska sitä voidaan hyödyntää myös muiden laitteiden kuin pelkästään prosessorin ja näytönohjaimen kanssa. CUDA on näistä kahdesta kuitenin parempi kvanttilaskennan ja muun suurta suoritustehoa vaativan laskennan suorittamiseen, sillä se on optimoitu toimimaan näytönohjaimen kanssa. OpenCL:n heikompi suoritusteho johtuu todennäköisesti sen yleiskäyttöisyydestä. Molemmat rajapinnat ovat kuitenkin tarkoitukseensa hyvin toimivia, ja mikäli ei tavoitella maksimaalista suoritustehoa, voidaan valinta näiden välillä tehdä rajapinnan tuntemisen ja käytössä olevan laitteiston perusteella (Karimi, Zilic & Radecka (2010)).

3.2 Simulointi supertietokoneilla

Toimintaperiaatteiltaan supertietokoneille suunnite ei juurikaan eroa edellä esitellystä CUDA-rajapintasimuloinnista: molemmissa laskenta hajautetaan useaan eri osaan, CUDA-rajapinnassa laskenta ositetaan näytönohjaimen multiprosessoreille, ja supertietokonesimulaatioissa puolestaan useille eri prosessoreille. Luonnollisesti supertietokoneilla voidaan mallintaa suurempia järjestelmiä, mutta niiden käyttömahdollisuudet ovat rajatumpia kuin erilaisten rajapintojen käyttäminen.

Supertietokoneilla laskentaa halutaan hajauttaa myös siitä syystä, että yhdellä prosessorilla pystytään mallintamaan vain noin 30-kubittisen järjestelmän toimintoja järkevästi. Ensimmäiset tällaiset simulaattorit toteutettiin vuonna 2002, jolloin sen aikaisella teknologialla mallinnettiin 30-kubittista järjestelmää. Nykyaikaisemmilla simulaattoreilla, kuten qHiPSTER-simulaattorilla, päästään supertietokoneesta riippuen 43-46 kubittiin (Smelyanskiy, Sawaya & Aspuru-Guzik (2016)).

qHiPSTER-simulaattorin testaamiseen käytettiin TACC Stampede -supertietokonetta, jonka 6400 laskentasolmusta simulaattorin käyttöön varattiin 1024. Tällöin simulaation käytössä on yli 32 teratavua muistia. Kuitenkin, huolimatta valtavasta laskentatehosta, ei tällä simulaattorilla voida suorittaa kvanttifourier-muunnoksia kuin noin 86 kappaletta vuorokaudessa 40-kubittisella järjestelmällä. Esimerkiksi Shorin algoritmissa tämä muunnos lasketaan jokaiselle tilalle, joten vaikka TACC Stampede oli artikkelin kirjoitushetkellä maailman kymmenenneksi tehokkain supertietokone, ei sillä voida ratkaista merkittäviä kvanttialgoritmeja (Smelyanskiy ym. (2016)).

Vaikka tietokoneiden laskentateho kehittyy koko ajan, arvioivat Smelyanskiy ym. (2016), että vuoteen 2022 mennessä qHiPSTERin avulla voidaan mallintaa vasta 49 kubittista järjestelmää. Tähän nostetaan syiksi paitsi muistintarpeen eksponentiaalinen kasvu, myös tiedonsiirtonopeuden rajallisuus laskentasolmujen välillä. Tilavektorin koon noustessa kasvaa siis paitsi sen laskemiseen kuluva aika, myös tiedonsiirto solmulta toiselle. Lisäksi osa operaatioista voidaan joutua suorittamaan useille prosessoreille yhtäaikaisesti, mikä myös lisää viivettä (Smelyanskiy ym. (2016)).

4 Emulointi

Miten kvanttijärjestelmien simulointi ja emulointi konkreettisesti eroavat toisistaan? Kuten aiemmin mainittiin, simulaation tavoitteena on toistaa mallinnettavan asian käytös ja emulaation tavoitteena puolestaan sisäinen toiminta (McGregor (2002)). Kvanttijärjestelmien osalta tämä tarkoittaa sitä, että simulaation täytyy laskea tarkasti jokainen yksittäinen operaatio, kun taas emulaation osalta riittää, että mallinnuksen lopputulos vastaa oikean kvanttijärjestelmän laskentatulosta. Tästä johtuen kvanttiemulaattoria voidaan tarvittaessa optimoida niissä kohdissa, missä esimerkiksi jokin kvanttioperaatio voidaan yhtäpitävästi korvata jollain klassisella operaatiolla (Häner ym. (2016)).

Kvanttijärjestelmien emuloinnille niiden simuloinnin sijaan on useita syitä. Esimerkiksi vuonna 2004 arvioitiin, että 20-kubitin järjestelmän yhden laskentatoimenpiteen simulointiin kuluisi kokonainen vuorokausi senaikaisilla tietokoneilla. Tietokoneiden laskentateho on noussut edellisen kymmenen vuoden aikana, mutta emuloimalla voidaan mallintaa kvanttilaskennan rinnakkaisuutta paremmin kuin simuloimalla. Lisäksi Feynman totesi jo vuonna 1982, että kvanttijärjestelmiä voidaan mallintaa tehokkaasti vain toisella kvanttitietokoneella (Feynman (1982), Khalid ym. (2004)). Tämä lausunto ei ole ristiriidassa aiemmin mainitun Gottesman-Knillin teoreeman kanssa, koska teoreema asettaa rajoitteita simuloitavalle järjestelmälle.

Tässä luvussa käydään läpi esimerkkejä kvanttijärjestelmien emuloinnista: Häner, Steiger, Smelyanskiy & Troyer (2016) tutkimusta kvanttipiirien emuloinnista supertietokoneen avulla ja Brian La Courin ym. tutkimuksia kokonaisen kvanttitietokoneen mallintamisesta klassisella järjestelmällä signaaliprosessoinnin avulla.

4.1 Kvanttitietokoneen emulointi signaaliprosessoinnilla

La Cour & Ott (2015) ovat kehittäneet QMT(*Quadrature Modulated Tonals*)-mallin, jonka avulla useamman kubitin järjestelmän toimintaa voidaan emuloida taajuuksia moduloimalla. Mallin pohjalta on toistaiseksi rakennettu toimiva kahden kubitin

järjestelmä(La Cour ym. (2016)), ja käytännön maksimimäärä nykyisellä tekniikalla on arviolta 40 kubittia, mutta teoriassa malli yleistyy mielivaltaiselle määrälle kubitteja. QMT-emuloinnissa jokaista järjestelmän 2^n amplitudia vastaa tietty taajuus. Merkittävä ero digitaaliseen simulointiin verrattuna on porttioperaatioiden suorittamisnopeus. Jokainen amplitudi voitaisi tallentaa tietokoneen muistiin, mutta porttioperaatioita vastaavat matriisioperaatiot jouduttaisi toteuttamaan peräkkäin, ja tämän aikavaativuus on 4^n, kun taas emuloidut porttioperaatiot suoritetaan yhtäaikaisesti (La Cour & Ott (2015)).

Mallin periaate on seuraavanlainen: kompleksiluku α voidaan kirjoittaa muodossa $\alpha = a + jb$, missä $a, b \in \mathbb{R}$. Nyt on mahdollista ilmaista kertoimet a ja b joko kahdella tasavirtasignaalin tai yhdellä vaihtovirtasignaalin taajuudella, sekä palauttaa kertoimet tästä signaalista palautuskaavojen avulla. Tämänkaltaista tekniikkaa käytetään nykyään esimerkiksi 64-QAM Ethernet -protokollassa, jossa 6-bittinen merkkijono esitetään 64 eri yhdistelmällä vaiheita ja amplitudeja. Yksi kubitti, eli $\alpha|0\rangle + \beta|1\rangle$, voidaan vastaavasti ilmaista kahdella kvadratuurisen signaalin taajuudella, ja porttioperaatiot toteutetaan muokkaamalla signaalia. Koska tässä lähestymistavassa ei tarvitse suorittaa matriisioperaatioita jokaiselle 2^n-komponenttiselle tilalle, päästään tällä tavalla lähemmäs aidon kvanttijärjestelmän toimintaa. Kubittien mittaukset suoritetaan vastaavanlaisella tavalla (La Cour & Ott (2015), La Cour ym. (2016)).

Nykyaikaisilla laitteilla pystytään muodostamaan sähköinen signaali taajuusalueelta 0,1 Hz - 100 GHz. Olettaen, että tämän signaalin kesto on 10 sekuntia, eli taajuusresoluutio on noin 0,1 Hz, voidaan tällä signaalilla mallintaa jopa teoreettisesti 40 kubitin toimintaa. Mikäli jokainen tähän signaaliin tallennettu lukupari tallennettaisiin tietokoneen muistiin, vaatisi se noin yhden teratavun verran tallennustilaa. Kvanttilaskennan vaativuudesta kertoo se, että 10 tunnin signaali vastaa 50 kubittia ja vuoden mittainen signaali 60 kubittia, ja vaikka signaalin kesto olisi koko universumin oletetun iän, vastaisi se silti vain noin 95 kubittia. Lisäksi, vaikka tätä vuoden mittaista signaalia tarkasteltaisiin Planckin vakion välein, vastaisi se siltikin vain 176 kubittista kvanttitietokonetta. Vertailun vuoksi, Shorin algoritmin tehokkaaseen toteutukseen esimerkiksi RSA-algoritmin purkamiseksi vaaditaan useita tuhansia

kubitteja. Tästä huolimatta jo tehokkaasti toimiva 40 kubittisen kvanttitietokoneen mallinnus olisi merkittävä kehitysaskel eteenpäin (La Cour & Ott (2015)).

Signaalin muodostaminen ei kuitenkaan ole tällä hetkellä ainoa rajoite, koska taajuussuodattimet asettavat oman rajoituksensa mallille. Nykyaikaisella puolijohdetekniikalla voidaan rakentaa piiri, jonka avulla voidaan teoriassa suodattaa edellämainittua taajuusväliä siten, että signaali vastaa 30 kubittista kvanttitietokonetta. 40 kubittisen koneen mallintaminen vaatii sekä taajuusalueen maksimin nostamista 1 THz asti, että tuhatkertaista transistoritiheyttä nykyisiin piireihin verrattuna (La Cour & Ott (2015)).

Tämänkaltaisen mallin eräs merkittävä etu supertietokoneilla ajettaviin mallinnuksiin verrattuna on sähköntarve. Siinä missä supertietokoneet vaativat useita megawatteja energiaa esimerkiksi 40 kubittisen simulaation ajamiseen, toimii QMT-malli vakioteholla riippumatta emuloitavien kubittien määrästä (La Cour & Ostrove (2017)).

4.2 Kvanttipiirien emulointi supertietokoneella

Supertietokoneella emulointi ei laitteiston osalta eroa juurikaan simuloinnista: myös emuloitaessa laskenta hajautetaan useiden laskentasolmujen kesken toimenpiteiden nopeuttamiseksi. Eroavaisuudet näiden mallintamistapojen välillä rinnakkaislaskennassa ovat siinä, mitä asioita lopulta lasketaan ja miten. Jos jokin laskutoimituksen osista voidaan ilmaista klassisessa muodossa, lasketaan se tällä tavalla ja täten koko laskenta nopeutuu merkittävästi. Simulaattorin täytyy puolestaan laskea jokainen yksittäinen kvanttisimulaaation vaihe täysin, sekä purkaa jokainen vaihe peruutettavissa oleviin kvanttiportteihin. Tällaisia tilanteita optimoimalla emuloinnilla saadaan merkittävästi simulointia nopeampaa (Häner ym. (2016)).

Häner ym. (2016) nostaa esiin neljä esimerkkiä operaatioista, joissa emulointi on kevyempi ratkaisu kuin simulointi: matemaattiset operaatiot, kvanttifourier-muunnos, kvanttivaiheen arviointi ja tuloksen mittaaminen. Näistä matemaattiset operaatiot ovat ehkä suoraviivaisin optimointikohde. Niiden kvanttisimulointi vaatii useiden

eri kvanttiporttien mallintamista, mutta emuloitaessa ne voidaan laskea suoraan klassisella tavalla. kvanttifourier-muunnosta voidaan puolestaan optimoida suorittamalla tilavektoreille nopea fouriermuunnos kvanttioperaatioiden laskemisen sijaan. Lisäksi Häner ym. (2016) mukaan kvanttiemulaattorilla on mittausvaiheessa merkittävä hyöty jopa fyysiseen kvanttitietokoneeseen verrattuna. Koska kvanttilaskenta on probabilistista, voi kvanttitietokone joutua suorittamaan algoritmin useita kertoja tulosjakauman saamiseksi. Vaikka emuloitaessa varsinainen laskenta on merkittävästi raskaampaa (aikavaativuus $O(2^n)$), näin saadaan kuitenkin yhdellä suorituksella koko jakauma (Häner ym. (2016)).

Emulaattorin ja simulaattorin nopeuserot esimerkiksi kvanttifourier-muunnoksen osalta ovat selkeitä. Yhtä solmua käytettäessä emulaattori on jopa viisitoista kertaa nopeampi kuin esimerkkikäytössä ollut simulaattori, kun järjestelmän koko on 28 kubittia. Matemaattisten operaatioiden osalta tehokkuusero on vielä suurempi. Koska Hänerin ym. emulaatiomallia ajettiin samalla TACC Stampede -supertietokoneella kuin aiemmin esiteltyä qHiPSTER-simulaatiomallia, voidaan näiden suoritustehoja vertailla keskenään. Kvanttifourier-muunnoksen osalta esimerkkikäytössä ollut simulaatiomalli on arviolta yhtä tehokas kuin qHiPSTER 28-kubittisen järjestelmän osalta, mutta kubittien määrän kasvaessa tehokkuusero kasvaa Hänerin ym. simulaattorin eduksi. Koska qHiPSTER on eräs tehokkaimmista simulaattoreista, ja Hänerin ym. simulaatiomalli on selvästi tätä tehokkaampi, on emulaatiomalli selvästi qHiPSTERiä tehokkaampi (Häner ym. (2016)).

5 Yhteenveto

Kuten tässä tutkielmassa on todettu, ei nykyisillä simulointi- tai emulointimenetelmillä pystytä mallintamaan kuin verrattain heikkotehoisia kvanttitietokoneita. Tästä huolimatta mallinnusmenetelmiä on tärkeä kehittää edelleen. Koska fyysisten kvanttitietokoneiden rakentaminen on sekä haastavaa että kallista, erilaisilla mallinnusmenetelmillä pystytään esimerkiksi kehitämään kvanttiohjelmointikieliä laajemman yhteisön toimesta.

Mallintamisen kehittyminen on ollut varsin nopeaa huomioiden sen, että kvanttitietokoneiden käsite on ylipäätään tunnettu vasta hieman yli kolmekymmentä vuotta. Tällöin Feynman esitti, ettei klassisella koneella ylipäätään voisikaan malllintaa kvanttijärjestelmien toimintaa. Gottesman-Knillin teoreeman myötä on kuitenkin osoitettu, että tietylle osajoukolle kvanttijärjestelmistä tämä on kuitenkin mahdollista. Jää kuitenkin nähtäväksi, että mitkä tällaisen osajoukon mahdollisuudet ovat, ja millaisen laitteiston niiden mallintaminen vaatii merkittävän laskettatehokkuuden nousun saamiseksi.

Tässä tutkielmassa esitetyistä mallintamiskeinoista potentiaalisimpina voidaan pitaa CUDA-rajapintaa ja muita GPU-ohjelmointirajapintoja tulevan tutkimuksen kannalta. Vaikka LaCourin QMT-emulointimallilla on saatu lupaavia tuloksia, mallin rakentamiseen vaadittavat osat eivät ole yhtä yleisesti saatavilla kuin tehokkaat näytönohjaimet. Lisäksi näytönohjainten laskentatehokkuus kasvaa edelleen jatkuvasti, joten uskon myös niillä simuloitavien järjestelmien kubittimäärien kasvavan. Täytyy kuitenkin muistaa, että jo yhden kubitin lisäys järjestelmään kaksinkertaistaa tilojen määrän edellisestä vaiheesta. Tämä paitsi vaikeuttaa laskutoimenpiteiden tekemistä, mutta myös lisää potentiaalista suoritustehoa.

GPU-rajapintojen avulla kenellä tahansa kvanttialgoritmeista ja kvanttitietokoneista kiinnostuneella on mahdollisuus opiskella aihetta. Tällä hetkellä tunnetuin yleisesti ohjelmoitavissa oleva fyysinen kvanttitietokone on viiden kubitin rekisterillä toimiva IBM QX, ja koska CUDA-rajapinnalla on onnistuttu simuloimaan jo yli kah-

denkymmenen kubitin rekisterillä toimivia algoritmeja, on tämän rajapinnan avulla mahdollista suorittaa huomattavasti vaativampia kvanttialgoritmeja.

Koska todennäköisesti ensimmäiset tehokkaat kvanttitietokoneet ovat vain yhden ongelman ratkaisuun suunniteltuja, niin siihen, että kvanttitietokoneet tulevat yleisesti saataville, kuluu todennäköisesti vielä vuosia. Erityisesti tästä syystä mallinnusmenetelmien kehittäminen on tärkeää. Kvanttitietokoneiden yleistyminen tulevaisuudessa on todennäköistä, ja kaikki tutkimustieto niiden mahdollisuuksista myös virtuaaliympäristössä nopeuttaa paitsi yleistymistä, myös tuo lisätietoa kvanttitietokoneiden mahdollisuuksista. Lisäksi on mahdollista, joskin epätodennäköistä, että jotakin merkittävää algoritmia onnistutaan mallintamaan klassisessa ympäristössä siten, että esimerkiksi salausalgoritmejä joudutaan kehittämään nykyistä turvallisemmiksi.

Kirjallisuutta

Cuffaro, M. E. (2015). *On the Significance of the Gottesman–Knill Theorem*. The British Journal for the Philosophy of Science, axv016.

Elhoushi, M., El-Kharashi, M., & Elrefaei, H. 2011. *Modeling a Quantum Processor using the QRAM Model*. Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference, IEEE

Feynman, R. *Simulating physics with computers*. International journal of theoretical physics 21.6 (1982): 467-488.

Gottesman, D. (1998). *The Heisenberg representation of quantum computers*. arXiv preprint quant-ph/9807006.

Grover, L. (1996). *A fast quantum mechanical algorithm for database search*. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (pp. 212-219). ACM.

Gutiérrez, E., Romero, S., Trenas, M. A., & Zapata, E. L. (2010). *Quantum computer simulation using the cuda programming model*. Computer Physics Communications, 181(2), 283-300.

Häner, T., Steiger, D. S., Smelyanskiy, M., & Troyer, M. (2016). *High performance emulation of quantum circuits*. arXiv preprint arXiv:1604.06460.

IBM Quantum Experience, *http://www.research.ibm.com/quantum*

Juliá-Díaz, B., Burdis, J. M., & Tabakin, F. (2006). *QDENSITY—a Mathematica quantum computer simulation. Computer Physics Communications*, 174(11), 914-934.

Karimi, K., Dickson, N. G., & Hamze, F. 2010. *A performance comparison of CUDA and OpenCL*. arXiv preprint arXiv:1005.2581.

Khalid, A, Zilic, Z & Radecka, K. 2004 *FPGA emulation of quantum circuits*. IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings., 2004, pp. 310-315. doi: 10.1109/ICCD.2004.1347938

La Cour, B.R. & Ostrove, C.I. 2017, *Subspace projection method for unstructured searches with noisy quantum oracles using a signal-based quantum emulation device*. Quantum Inf Process 16: 7. doi:10.1007/s11128-016-1464-z

La Cour, B.R, Ostrove, C.L, Ott, G.E., Starkey, M.J. & Wilson, G.R. 2016. *Classical emulation of a quantum computer*. International Journal of Quantum Information, 14, 1640004.

La Cour, B.R. & Ott, G.E. 2015. *Signal-based classical emulation of a universal quantum computer*. New Journal of Physics, 17, 053017.

McGregor, I. 2002. *The relationship between simulation and emulation*. Simulation Conference, 2002. Proceedings of the winter. IEEE

Nielsen, M. A., & Chuang, I. (2002). *Quantum computation and quantum information.*

Obenland, K. M., & Despain, A. M. (1998). *A parallel quantum computer simulator*. arXiv preprint quant-ph/9804039.

Rønnow, T. F., Wang, Z., Job, J., Boixo, S., Isakov, S. V., Wecker, D., Martinis, J., Lidar, D. & Troyer, M. (2014). *Defining and detecting quantum speedup.* Science, 345(6195), 420-424.

Selinger, P. 2004. *A brief survey of quantum programming languages*. Functional and Logic Programming, 61-69.

Shin, S. W., Smith, G., Smolin, J. A., & Vazirani, U. (2014). *How"Quantum"is the D-Wave Machine?.* arXiv preprint arXiv:1401.7087.

Shor, P. W. (1999). *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*. SIAM review, 41(2), 303-332.

Smelyanskiy, M., Sawaya, N. P., & Aspuru-Guzik, A. (2016). *qHiPSTER: the quantum high performance software testing environment.* arXiv preprint arXiv:1601.07195.