Automatsko prevođenje izvornog koda za programske jezike klasičnog računala u izvorni kod za programske jezike kvantnog računala

Bojić, Alan

Sveučilište u Zagrebu

Fakultet organizacije i informatike

Alan Bojić

AUTOMATSKO PREVOĐENJE IZVORNOGA KODA ZA PROGRAMSKE JEZIKE KLASIČNOGA RAČUNALA U IZVORNI KOD ZA PROGRAMSKE JEZIKE KVANTNOGA RAČUNALA

DOKTORSKI RAD

Mentori:
Prof. dr. sc. Alen Lovrenčić
Doc. dr. sc. Ivan Hip

Varaždin, 2018.

Sveučilište u Zagrebu

Faculty of Organization and Informatics

Alan Bojić

AUTOMATIC CONVERSION OF SOURCE CODE OF PROGRAMMING LANGUAGES FOR CLASSICAL COMPUTER INTO SOURCE CODE OF PROGRAMMING LANGUAGES FOR QUANTUM COMPUTER

DOCTORAL THESIS

Supervisors:
Alen Lovrenčić, PhD, Professor
Ivan Hip, PhD, Assistant Professor

Varaždin, 2018

PODACI O DOKTORSKOM RADU

I. AUTOR

Ime i prezime	Alan Bojić
Datum i mjesto rođenja	07. studeni. 1984., Zagreb
Naziv fakulteta i datum diplomiranja	Fakultet organizacije i informatike, 15. travnja 2008.
Sadašnje zaposlenje	Arhitekt programskih rješenja, Agramsoft, Zagreb.

II. DOKTORSKI RAD

Naslov	AUTOMATSKO PREVOĐENJE IZVORNOGA KODA ZA PROGRAMSKE JEZIKE KLASIČNOGA RAČUNALA U IZVORNI KOD ZA PROGRAMSKE JEZIKE KVANTNOGA RAČUNALA
Broj stranica, slika, tabela, priloga, bibliografskih podataka	160 stranica, 42 slike, 24 tablice, 22 grafikona, 115 bibliografskih podataka
Znanstveno područje i polje iz kojeg je postignut akademski stupanj	Društvene znanosti, informacijske i komunikacijske znanosti
Mentor i voditelj rada	Prof. dr. sc. Alen Lovrenčić Doc. dr. sc. Ivan Hip
Fakultet na kojem je rad obranjen	Fakultet organizacije i informatike
Oznaka i redni broj rada	140

III. OCJENA I OBRANA

Datum sjednice Fakultetskog vijeća na kojoj je prihvaćena tema	25. travnja 2017.
Datum predaje rada	02. listopada 2017.
Datum sjednice Fakultetskog vijeća na kojoj je prihvaćena pozitivna ocjena rada	16. svibnja 2018.
Sastav Povjerenstva koje je rad ocijenilo	Izv. prof. dr. sc. Ivan Magdalenić prof. dr. sc. Mirko Maleković prof. dr. sc. Robert Manger
Datum obrane	07. lipnja 2018.
Sastav Povjerenstva pred kojim je rad obranjen	Izv. prof. dr. sc. Ivan Magdalenić prof. dr. sc. Mirko Maleković prof. dr. sc. Robert Manger
Datum promocije	

Zahvala

Zahvaljujem se…

Roditeljima na neizmjernoj podršci u svemu što sam radio u životu pa tako i u izradi ovog rada.

Mentorima prof. dr. sc. Alenu Lovrenčiću i doc. dr. sc. Ivanu Hipu na uloženom velikom trudu, brojnim savjetima i kvalitetnom vođenju kroz cijeli proces istraživanja.

Članovima povjerenstva doc. dr. sc. Ivanu Magdaleniću, prof. dr. sc. Mirku Malekoviću i prof. dr. sc. Robert Mangeru na vrlo vrijednim i kvalitetnim savjetima koji su značajno pridonijeli kvaliteti konačnog sadržaja ovog rada.

FOI-u kao instituciji koja mi je pružila životno obrazovanje i iskustvo.

IN2 d.o.o. kao firmi koja me podržala i izlazila u susret kada god je to bilo potrebno.

Svim prijateljima koji su me moralno poticali da završim ovo istraživanje.

Gradu Varaždinu na gostoprimstvu i nezaboravnim studenskim danima.

Sažetak

Kvantno računalo koristi kvantnomehaničke efekte poput kvantne prepletenosti i kvantne superpozicije prilikom obrade informacija. Ukoliko jednoga dana kvantno računalo zaista postane komercijalni proizvod, pojavit će se potreba za automatskim prevođenjem izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala zbog toga što će takav pristup ubrzati tranziciju postojećih aplikacija te će ujedno omogućiti razvojnim programskim inženjerima pisanje programa za kvantno računalo u programskim jezicima za klasično računalo. U ovome radu bit će opisan, implementiran i testiran prevoditelj izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala. U sklopu istraživanja mjerit će se točnost generiranog izvornoga koda kvantnih računala te kvantni trošak kvantnih sklopovlja koji su generirani upotrebom QR i CS algoritama te upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno na skupu podataka koji predstavlja programe klasičnoga računala.

Ključne riječi: kvantno računalo, klasično računalo, programski jezici, prevođenje izvornog koda

Abstract

Quantum computers use quantum mechanical effects such as quantum entanglement and quantum superposition. If one day the quantum computer becomes a commercially available product, then it is very important to have a transcompiler that will convert the source code of programming languages for classical computers into the source code of programming languages for quantum computers because it will speed up the translation process of existing applications and ensure that developers can write programs for quantum computers using programming languages for classical computers. This research proposes, implements and tests an approach for the conversion of the source code of programming languages for classical computers into the source code of programming languages for quantum computers. The accuracy of the generated source code for quantum computers and the quantum cost of quantum circuits which are generated using QR, CS algorithms and algorithm for direct translation of reversible circuit into quantum circuit will be measured on a data set which represents programmes on classical computers.

Keywords: quantum computer, classical computer, programming languages, source code conversion

Sadržaj

Popis slika

Popis tablica

Popis grafikona

1. Uvod

1.1. Povijest kvantnoga računarstva

Prema Mooreovom zakonu, broj tranzistora u računalnim procesorima udvostručuje se za konstantnu cijenu svake dvije godine [83][109]. To je pravilo opisivalo razvoj računalnih procesora u proteklim desetljećima, međutim zbog temeljnih problema u današnjoj tehnologiji proizvodnje računalnih procesora taj zakon prestaje vrijediti. Problem današnje tehnologije je taj da komponente računalnih procesora postaju sve manje, što za posljedicu ima utjecaj kvantnomehaničkih efekata na njihov rad. Jedna od mogućih alternativnih paradigmi klasičnom računarstvu je paradigma znanosti o kvantnoj informaciji, kojoj je cilj iskoristiti kvantnomehaničke efekte u obradi, manipulaciji i prijenosu informacija za razliku od današnjih elektroničkih uređaja, čije se funkcioniranje temelji na klasičnoj teoriji informacija. U teoriji, model kvantnoga računala prepoznat je kao potencijalno bolji model za obradu informacija od modela klasičnoga računala, što će biti objašnjeno kasnije u radu.

Kvantno računarstvo je znanstveno područje koje se bavi proučavanjem upotrebe kvantnomehaničkih efekata, poput superpozicije i prepletenosti, prilikom obrade informacija. To je vrlo mlado i neistraženo znanstveno područje čija ću značajnija postignuća spomenuti u nastavku.

Richard Feynman, poznati američki teorijski fizičar, koji je sudjelovao u razvoju atomske bombe tijekom Drugog svjetskog rata, smatra se pionirom kvantnoga računarstva. Feynman je osamdesetih godina dvadesetog stoljeća počeo istraživati kako simulirati kvantne sustave pomoću drugih kvantnih sustava, a ne pomoću klasičnoga računala. Proučavao je kako realizirati koncepte klasičnih računala, poput binarnih brojeva, upotrebom kvantnih sustava [31].

Britanski fizičar David Deutsch je 1985. godine opisao koncepte univerzalnog kvantnoga računala. Deutsch je pokušao pokazati da je upotrebom univerzalnog kvantnoga računala moguće simulirati bilo koji fizički sustav [22], te je opisao koncept kontroliranih jednostavnih operacija nad kvantnomehaničkim sustavom kojima je moguće utjecati na evoluciju tog sustava. Kasnije u radu ćemo vidjeti da su te jednostavne operacije u biti kvantna vrata koja su analogna logičkim vratima kod klasičnih računala. Danas Deutschov model predstavlja osnovu za izradu kvantnih algoritama i proučavanje računalne snage

kvantnih računala. Deutsch je osim samog modela univerzalnog kvantnoga računala predstavio i niz kvantnih algoritama kojima je ukazao na računsku moć modela kvantnoga računala. Jedan od takvih algoritama je i Deutsch–Jozsa algoritam, kojeg je objavio zajedno s Richardom Jozsaom 1992. godine. Algoritam odgovara na pitanje je li funkcija balansirana [21] te je zanimljiv zbog toga što ima konstantnu složenost u usporedbi s algoritmom za klasična računala koji za isti problem ima linearnu složenost.

Peter Shor, profesor primijenjene matematike na MIT-u, 1994. godine objavljuje algoritam za kvantna računala koji faktorizira cijeli broj na primarne faktore [96]. Shorov algoritam je značajno brži od najbržeg poznatog algoritma za isti problem na klasičnim računalima te se danas smatra najznačajnijim algoritmom za kvantna računala jer rješava bitan problem u računarstvu, te ukazuje da bi kvantna računala mogla imati značajno više računske snage od klasičnih računala. Zanimljivost vezana uz Shorov algoritam je da on dovodi u pitanje Church-Turingovu tezu, prema kojoj je bilo koji algoritam za bilo koje fizički izvedivo računalo moguće simulirati na Turingovom stroju uz maksimalno polinomno usporavanje [102], što sa Shorovim algoritmom, barem za sada, nije slučaj.

Lov Grover je 1996. godine objavio kvantni algoritam za pretraživanje nesortirane liste elemenata [38]. Taj problem je prisutan u današnjem računarstvu zbog velikih količina nesortiranih podataka u današnjim bazama podataka. Groverov algoritam pronalazi element u nesortiranoj listi brže nego algoritam za klasična računala te na taj način ukazuje na računsku snagu kvantnoga računala naspram klasičnoga računala u rješavanju još jednog vrlo bitnog problema u računarstvu. Danas je kvantno računarstvo predmet istraživanja mnogih znanstvenika, a vlade mnogih država ulažu značajna sredstva u ovo područje [82][12].

1.2. Povijest kvantnoga računala

Kvantno računalo je uređaj za obradu informacija koji prilikom izvršavanja operacija nad podacima koristi kvantnomehaničke efekte poput superpozicije i prepletenosti [43]. Osim teorijskih aspekata vezanih uz kvantno računarstvo kojima se bave teorijski fizičari, eksperimentalni fizičari pokušavaju izgraditi prvo kvantno računalo većih razmjera. Veliki problem u izgradnji kvantnoga računala većih razmjera predstavlja kvantna dekoherencija, odnosno utjecaj okoline na kvantno računalo. Kvantna dekoherencija narušava kvantnu

informaciju unutar računala [43][109]. U nastavku ću kronološki navesti značajnija postignuća vezana uz fizičku realizaciju kvantnoga računala.

Vandersypen, Steffen, Breyta, Yannoni, Sherwood i Chuang su 2001. godine uspjeli demonstrirati Shorov algoritam na kvantnom računalu sa 7 kvantnih bitova koji su bili konstruirani pomoću tehnologije nuklearne magnetske rezonance [106].

2005. godine skupina znanstvenika sa Sveučilišta u Michiganu uspjela je konstruirati kvantnomehanički čip koji se temelji na tehnologiji ionske zamke [49].

DiCarlo, Chow, Gambetta, Bishop, Johnson, Schuster, Majer, Blais, Frunzio, Girvin i Schoelkopf su 2009. godine uspjeli demonstrirati kvantne algoritme na kvantnom procesoru s dva kvantna bita [24].

2010. godine tim znanstvenika sa Sveučilišta u Bristolu uspio je kreirati kvantni čip temeljen na kvantnoj optici te na njemu demonstrirati Shorov algoritam [87].

2011. godine tvrtka D-Wave Systems je stavila na tržište prvo komercijalno kvantno računalo naziva „D-Wave One“. Radi se o kvantnom procesoru sa 128 kvantnih bitova, čija cijena iznosi 10 milijuna dolara [53]. Oko tog procesora je bilo dosta polemike vezano uz pitanje je li taj procesor ujedno i univerzalno kvantno računalo. Tim predvođen Matthiasom Troyerom i Danielom Lidarom došao je do zaključka da procesor upotrebljava određene kvantnomehaničke efekte, međutim ne rješava probleme brže u odnosu na klasično računalo [6]. Danas je već dostupan i nasljednik „D-Wave One“ procesora, „D-Wave Two“.

2011. godine skupina znanstvenika je uspjela konstruirati kvantno računalo temeljeno na Von Neumannovoj arhitekturi [6].

Međunarodna skupina znanstvenika je 2012. godine konstruirala dvo-qubitno kvantno računalo unutar kristala dijamanta. Specifičnost te tehnologije je da se vrlo lako može skalirati na sobnoj temperaturi [104].

Krajem 2012. godine australski znanstvenici su uspjeli realizirati kvantni bit pomoću jednog atoma [33], a u Vancouveru je osnovana prva programska tvrtka 1QBit koja proizvodi programska rješenja za kvantna računala uključujući i programska rješenja za „D-Wave Two“ procesor [41].

Google je u svibnju 2013. godine osnovao laboratorij za kvantnu umjetnu inteligenciju, opremljen kvantnim računalom s 512 kvantnih bitova proizvođača D-Wave. Svrha laboratorija je okupiti znanstvenike diljem svijeta koji za cilj imaju poboljšati računalno učenje upotrebom kvantnoga računala [47].

Početkom 2014. godine objavljeno je da je Nacionalna sigurnosna agencija SAD-a pokrenula projekt vrijedan 79,7 milijuna dolara, kojemu je cilj sagledati upotrebu kvantnoga računala u razbijanju kriptiranih podataka [59].

Krajem 2015. godine NASA je objavila da posjeduje kvantno računalo u vrijednosti od 15 milijuna dolara koje je sastavila kompanija D-Wave Systems.

U 2016. godini znanstvenici sa Sveučilišta u Baselu opisali su koncept kvantnoga računala koji se temelji na elektroničkim šupljinama, a koji bi trebao omogućiti izradu kvantnoga računala još većih razmjera [88].

Sredinom 2017. godine IBM je objavio da je uspješno napravio i testirao komercijalni univerzalni kvantni procesor s 17 kvantnih bitova.

Krajem 2017. godine Microsoft je izdao skup razvojnih alata za kvantna računala – „Quantum Development Kit“ [61].

Google je u ožujku 2018. godine objavio novi kvantni procesor s 72 kvantna bita naziva „Bristlecone“.

1.3. Problem automatskog prevođenja izvornoga koda

Danas postoje brojne aplikacije i programi pisani u starijim tehnologijama koje je potrebno prilagoditi novim tehnologijama, što nije nimalo jednostavno i iziskuje mnogo resursa. Tipičan primjer iz prakse je prevođenje izvornoga koda u Oracle Forms tehnologiji, gdje se često javlja potreba da se izvorni kod iz starije verzije tehnologije Oracle Forms (Oracle Forms 6i) prevede u izvorni kod novije Oracle Forms tehnologije (Oracle Forms 12g). Taj problem je u praksi riješen upotrebom Oracleovog prevoditelja *(eng. transcompiler)*. Prevoditelji su programska rješenja koja prevode izvorni kod jednog programskog jezika u izvorni kod drugog programskog jezika. Benefiti prevoditelja su u tome da se aplikacije i programi ne moraju ponovo pisati, što pridonosi značajnoj uštedi resursa. Među ostalim,

prevoditelji olakšavaju debugiranje te omogućuju i automatsko refaktoriranje koda. Neki od prevoditelja koji se pojavljuju u praksi su j2objc (Java u Objective-C) [60], Script# (C# u Javascript) [51], p3cobol (COBOL u Javu) [57] te Spark (C u VHDL) [48].

Do danas ne postoji niti jedan dostupan prevoditelj izvornoga koda klasičnoga računala u izvorni kod kvantnoga računala, tako da će izrada prvog takvog prevoditelja biti izazov, pogotovo zbog toga što je logika kvantnih računala značajno drugačija od logike klasičnih računala, što će biti detaljnije objašnjeno kasnije u radu.

1.4. Problem testiranja dekompozicijskih algoritama

Kao što je već spomenuto, eksperimentalni fizičari pokušavaju konstruirati kvantno računalo većih razmjera, dok teorijski fizičari i matematičari pokušavaju konstruirati kvantne algoritme koji rješavaju probleme značajno brže od klasičnoga računala. U konstruiranju kvantnih algoritama jako bitnu ulogu imaju dekompozicijski algoritmi unitarnih matrica. To su algoritmi koji za ulaz uzimaju unitarnu matricu, dok za izlaz imaju niz jednostavnijih unitarnih matrica manjih dimenzija koje će u konačnici biti realizirane pomoću specifičnih kvantnih vrata. Dakako, cilj svakog dekompozicijskog algoritma je generirati ukupno brojčano što manje i dimenzijama što jednostavnije unitarne matrice. Dekompozicijski algoritmi će biti detaljnije u poglavlju 6.1.2. Do danas ne postoji specifičan okvir za testiranje dekompozicijskih algoritama. Takav okvir bi trebao omogućiti jednostavno uključivanje novih algoritama te definiranje kriterija za testiranje. Također, okvir bi trebao pružiti i odgovarajući skup podataka za testiranje dekompozicijskih algoritama.

2. Motivacija za istraživanje

Kronologija značajnijih postignuća vezanih uz kvantno računarstvo i kvantna računala ukazuje na to da je već sada računalna snaga kvantnoga računala prepoznata diljem svijeta te da se s vremenom sve više resursa ulaže u njegovo razvijanje. Ukoliko jednoga dana kvantno računalo zaista i postane komercijalni proizvod, razvoj novih kao i prebacivanje postojećih programskih rješenja bit će izazov za IT sektor. Pojavit će se potreba za automatskim prevođenjem izvornoga koda programskih jezika za klasična računala poput Java, C, C++, C#, VHDL u izvorni kod programskih jezika za kvantna računala poput QCL [84], QPL [94] i LanQ [79]. Osim prevođenja izvornoga koda, veliki izazov razvojnim inženjerima klasičnih računala će biti savladavanje logike kvantnih računala, s obzirom da je ta logika značajno drugačija od logike klasičnih računala. Prevoditelj koji bi pretvarao izvorni kod klasičnih računala u izvorni kod kvantnih računala bi značajno olakšao taj prijelaz jer bi omogućio programiranje kvantnih računala u programskim jezicima klasičnih računala. Sve dok se ne pojavi stvarna potreba za takvim prevoditeljem, prevoditelj može poslužiti u edukacijske svrhe te za testiranje algoritama za dekompoziciju unitarnih matrica, što je vrlo bitno za daljnji razvoj područja. Navedeni razlozi bili su dovoljni da me potaknu na razmišljanje je li i na koji način moguće napraviti prevoditelja koji pretvara izvorni kod klasičnih računala u izvorni kod kvantnih računala. Osim same implementacije prevoditelja, dodatna motivacija mi je bila i usporedba algoritama za dekompoziciju unitarnih matrica nad skupom unitarnih matrica koje opisuju programe na klasičnim računalima.

3. Hipoteza i ciljevi istraživanja

Hipoteze:

H1. Na klasičnom računalu moguće je uspješno implementirati automatsko prevođenje izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala

Podhipoteze su:

H1A. Generirano kvantno sklopovlje kojeg implementira generirani izvorni kod programskog jezika za kvantno računalo imat će kvantni trošak koji je manji ili jednak kvantnom trošku reverzibilnog sklopovlja koje je generirano na temelju izvornoga koda za programski jezik klasičnoga računala

H1B. Generirani izvorni kod programskog jezika za kvantno računala koji implementira generirano kvantno sklopovlje koje je generirano na temelju izvornoga koda za programski jezik klasičnoga računala imat će 100%-tnu točnost

Ciljevi:

C1. Prvi cilj istraživanja je osmisliti pristup za automatsko prevođenje izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala

C2. Drugi cilj istraživanja je implementirati prevoditelja koji se temelji na osmišljenom pristupu

C3. Treći cilj istraživanja je implementirati okvir za testiranje algoritama za generiranje kvantnih sklopova

C4. Četvrti cilj istraživanja je testirati implementiranog prevoditelja s gledišta točnosti generiranog koda i kvantnoga troška kvantnih sklopovlja koji su generirani upotrebom QR, CS algoritama te upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno na skupu podataka koji predstavlja programe klasičnoga računala

3.1. Plan istraživanja

Istraživanje će biti podijeljeno u četiri dijela:

1. Osmišljavanje teorijskog pristupa za prevođenje izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala.

U ovom dijelu istraživanja koristit će se metoda kompilacije, misaonog eksperimenta i logičkog modeliranja kako bi se postigao prvi istraživački cilj, a to je osmišljavanje pristupa za automatsko prevođenje izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala. Ovaj dio istraživanja se sastoji od dva koraka:

a) Istraživanje literature

U ovom koraku istražuje se literatura (metoda kompilacije) vezana uz pojedine korake prevođenja u svrhu izgradnje misaonog modela (misaonog eksperimenta) prevođenja izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala.

b) Definiranje teorijskog modela za prevođenje izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala

U drugom koraku prvog dijela istraživanja se na temelju istražene literature i misaonog modela definira logički model pretvorbe logike klasičnoga računala u logiku kvantnoga računala. Pretvorba izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala temeljit će se na prikazu logike izvornoga koda za programske jezike klasičnoga računala preko tablice istinitosti koja će se proširiti tako da bude reverzibilna [27]. Na temelju reverzibilne tablice istinitosti generirat će se reverzibilno sklopovlje [77]. Reverzibilno sklopovlje pretvarat će se u kvantno sklopovlje upotrebom algoritama za generiranje kvantnih sklopova (QR, CS, Izravna pretvorba reverzibilnog sklopovlja u kvantno). Generirano kvantno sklopovlje pretvorit će se u izvorni kod za programske jezike kvantnoga računala.

2. Implementacija prevoditelja izvornoga koda za programski jezik klasičnoga računala u izvorni kod za programski jezik kvantnoga računala na temelju osmišljenog pristupa

U ovom dijelu istraživanja implementirat će se prevoditelj izvornoga koda za programski jezik klasičnoga računala u izvorni kod za programski jezik kvantnoga računala na temelju pristupa osmišljenog u prvom dijelu istraživanja. U ovom dijelu istraživanja bit će realiziran drugi i treći cilj istraživanja.

a) Tehnologija

Prevoditelj će biti implementiran u C++ programskom jeziku iz razloga što se programi napisani u C++ tehnologiji brzo izvršavaju na računalu, a to je vrlo bitno jer će prevoditelj izvršavati vrlo zahtjevne linearne operacije. Pored toga, za C++ programski jezik postoji dosta dostupnih biblioteka za linearnu algebru i reverzibilna sklopovlja (npr. LAPACK, Armadillo, RevKit). Izvorni programski jezik za klasično računalo je VDHL, dok je odredišni jezik za kvantno računalo QCL.

b) Implementacija

U ovom koraku će se metodom konkretizacije provesti implementacija definiranog logičkog modela pretvorbe logike klasičnoga računala u logiku kvantnoga računala u C++ programskom jeziku. U sklopu toga bit će implementiran i okvir za testiranje algoritama za generiranje kvantnih sklopova.

3. Testiranje implementiranog prevoditelja i algoritama za generiranje kvantnih sklopova

U ovom djelu istraživanja koristit će se eksperimentalna metoda za provođenje eksperimenata nad implementiranim prevoditeljem na skupu podataka koji predstavlja programe klasičnoga računala. Ograničenje u provođenju eksperimenata bit će uvjetovano performansama računala u smislu da će performanse računala ograničiti skup eksperimenata koje je moguće provesti u prihvatljivom vremenu (jedan eksperiment unutar 7200 sekundi).

a) Generator izvornoga koda

Za potrebe testiranja implementiranog prevoditelja bit će potrebno implementirati generator izvornih kodova za VHDL programski jezik koji implementiraju nasumičnu tablicu istinitosti.

b) Provođenje testiranja implementiranog prevoditelja

U ovom koraku mjerit će se točnost generiranog koda te kvantni trošak reverzibilnih i kvantnih sklopova koji su generirani upotrebom QR, CS algoritama te upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno na skupu podataka koji je generiran u prethodnom koraku. Točnost generiranog koda mjerit će se preko vjerojatnosti točnih izlaza koja će se izračunati na temelju elemenata vektora stanja kvantnoga sustava unutar QCL simulatora nakon izvršenja generiranog QCL izvornoga koda. Kvantni trošak reverzibilnih i kvantnih sklopovlja će se računati preko broja i složenosti vrata od kojih se generirani sklopovi sastoje.

4. Tumačenje dobivenih rezultata testiranja i donošenje zaključaka

a) Tumačenje dobivenih rezultata testiranja i donošenje zaključka

U ovom koraku koristit će se metoda komparacije kako bi se usporedili kvantni troškovi reverzibilnih i kvantnih sklopova koji su generirani u pojedinim testovima. Također, metodom komparacije će se usporediti i točnost kvantnih sklopova koji su generirani u svakom pojedinačnom testu u odnosu na postavljeni prag u H1B podhipotezi. Na temelju provedenih usporedbi rezultata svih provedenih testova dat će se odgovor na postavljenu hipotezu.

4. Društveni i znanstveni doprinos rada

Društveni i znanstveni doprinos rada očituje se u nekoliko područja te je opisan u nastavku.

4.1. Uvod u automatsko prevođenje izvornoga koda klasičnih računala u izvorni kod kvantnih računala

Predloženi pristup prevođenju izvornoga koda programskih jezika klasičnih računala u izvorni kod programskih jezika kvantnih računala je ujedno i prvi takav pristup koji je službeno objavljen do vremena izdavanja ovog rada. Smatram da opisani pristup budi svijest o važnosti automatskog prevođenja izvornoga koda između klasičnih i kvantnih računala te da predstavlja dobru osnovu za daljnja istraživanja vezana uz tu problematiku zato što postoje mjesta za daljnja poboljšanja.

4.2. Prevođenje izvornoga VHDL koda u izvorni QCL kod

Implementirani prevoditelj koji se temelji na predloženom pristupu omogućuje automatsko prevođenje izvornoga VHDL koda u izvorni QCL kod, za čim će se možda ukazati potreba u budućnosti. Jedan od motiva za prevođenje VHDL koda u QCL kod može biti posljedica želje da se program pisan u VHDL kodu izvrši na kvantnom računalu koje „razumije“ QCL kod ili pak želja da se program pisan u VHDL programskom jeziku nastavi održavati u QCL programskom jeziku.

4.3. Programiranje kvantnoga računala upotrebom VHDL programskog jezika

Implementirani prevoditelj koji se temelji na predloženom pristupu omogućuje da razvojni inženjeri programiraju kvantno računalo upotrebom VHDL programskog jezika za klasična računala.

4.4. Okvir za usporedbu algoritama za dekompoziciju unitarnih matrica nad skupom unitarnih matrica koje predstavljaju programe na klasičnom računalu

U sklopu implementacije prevoditelja bit će realiziran i okvir koji će omogućiti testiranje algoritama za dekompoziciju unitarnih matrica nad skupom unitarnih matrica koje predstavljaju programe klasičnih računala.

4.5. Usporedba QR i CS algoritama za dekompoziciju unitarnih matrica nad skupom unitarnih matrica koje predstavljaju programe na klasičnom računalu

U literaturi QR [34] i CS [101] algoritmi su među najčešće spominjanim algoritmima za dekompoziciju proizvoljnih unitarnih matrica prilikom generiranja kvantnog sklopovlja. U ovom radu će navedeni algoritmi biti testirani nad specifičnim skupom podataka, a to je skup unitarnih matrica koje predstavljaju programe klasičnih računala. Cilj je pokazati koji je od ta dva algoritma pogodniji za korištenje pri prevođenju izvornoga koda klasičnih računala u izvorni kod kvantnih računala.

5. Osnove kvantnoga računarstva

U ovom dijelu rada bit će opisani osnovni koncepti koji su ključni za razumijevanje logike kvantnoga računala, pa tako i nastavka rada.

5.1. Uvod u kvantnu mehaniku

Kvantna mehanika je grana fizike koja proučava ponašanje elementarnih čestica u atomima, molekulama, kristalima i atomskim jezgrama [45][64][70][83]. Postulati kvantne mehanike predstavljaju sponu između fizičkog svijeta i matematičkog formalizma kvantne mehanike. U kvantnoj mehanici koristi se Diracova *bra-ket* notacija u kojoj se vektor u vektorskom prostoru označava s $|\psi\rangle$ [70][83]. ψ je naziv vektora, dok notacija $|.\rangle$ označava da je objekt vektor. Cijeli objekt $|\psi\rangle$ se u literaturi ponekad označava i s *ket*, dok notacijom $\langle\psi|$ označavamo dualni vektor vektora $|\psi\rangle$. Tablica 1. prikazuje standardne notacije u kvantnoj mehanici [83]. U nastavku rada opisani su postulati na kojima se temelji kvantna mehanika.

Tablica 1: Diracova notacija pojmova iz linearne algebre

Notacija	Opis
z^*	Kompleksno konjugirani kompleksni broj z.
$\|\psi\rangle$	Vektor *ket*.
$\langle\psi\|$	Dualni vektor vektora $|\psi\rangle$. U literaturi se još naziva *bra*.
$\langle\varphi\|\psi\rangle$	Skalarni produkt vektora $|\varphi\rangle$ i $|\psi\rangle$.
$\|\varphi\rangle \otimes \|\psi\rangle$	Tenzorski produkt vektora $|\varphi\rangle$ i $|\psi\rangle$.
$\|\varphi\rangle\|\psi\rangle$	Još jedan način prikaza tenzorskog produkta vektora $|\varphi\rangle$ i $|\psi\rangle$.
A^*	Kompleksno konjugirana matrica A.
A^T	Transponirana matrica A.
$A^\dagger$	Hermitski konjugirana matrica A.
$\langle\varphi\|A\|\psi\rangle$	Skalarni produkt između $|\varphi\rangle$ i $A|\psi\rangle$.

5.1.1. Prostor stanja

Prvi postulat kvantne mehanike definira prostor u kojem se prikazuje stanje kvantnoga sustava. Taj prostor se naziva Hilbertov prostor [70][83].

Postulat 1: Prostor koji je povezan s bilo kojim izoliranim fizičkim sustavom je kompleksni vektorski prostor s definiranim skalarnim produktom (Hilbertov prostor) te se

naziva prostorom stanja sustava. Sustav je potpuno opisan s njegovim vektorom stanja koji je jedinični vektor u prostoru stanja sustava [83].

Prvi postulat kvantne mehanike nam ne govori o tome koji konkretno prostor stanja treba koristiti za neki fizički sustav, niti koji je njegov vektor stanja u tom prostoru. Da bi se definirao prostor i vektor stanja nekog sustava potrebno je definirati mnogo pravila koja opisuju određeni sustav.

5.1.2. Evolucija kvantnomehaničkog sustava

Postulat o evoluciji kvantnomehaničkog sustava govori o tome kako se stanje kvantnoga sustava $|\psi\rangle$ mijenja u ovisnosti o vremenu [70][83].

Postulat 2: Evolucija zatvorenog kvantnoga sustava je opisana unitarnom transformacijom, tj. stanje sustava $|\psi\rangle$ u vremenu t_1 je u odnosu sa stanjem sustava $|\psi'\rangle$ u vremenu t_2 preko unitarnog operatora U koji ovisi o t_1 i t_2 [83]:

$$|\psi'\rangle = U|\psi\rangle.$$

Kao što nam kvantna mehanika ne govori koji prostor stanja koristiti, odnosno koje je stanje kvantnoga sustava, tako nam ne govori ni koji unitarni operator koristiti.

Unitarni operatori se mogu prikazati pomoću unitarnih matrica. Unitarna matrica U ima svojstvo

$$U^{\dagger}U = I.$$

Drugi postulat kvantne mehanike zahtijeva da sustav bude zatvoren, odnosno da nema interakciju s vanjskim svijetom. Dakako, to je u praksi gotovo nemoguće zato što je svaki sustav dio nekog većeg sustava i u stalnoj je interakciji s njim. Upravo ta činjenica, da je gotovo nemoguće stvoriti savršeno izoliran sustav, ugrožava fizičku realizaciju kvantnoga računala jer interakcija s vanjskim sustavom (kvantna dekoherencija) narušava kvantnu informaciju koja je pohranjena u samom kvantnom sustavu. Dok drugi postulat kvantne mehanike opisuje diskretno u vremenu odnos stanja kvantnoga sustava, Ervin Schrödinger je svojom poznatom jednadžbom opisao i kontinuiranu evoluciju kvantnoga sustava [91]

$$i\hbar \frac{d|\psi\rangle}{dt} = H|\psi\rangle,$$

gdje je $\hbar$ reducirana Planckova konstanta, a H hamiltonijan sustava. Hamiltonijan sustava je najvažnija komponenta Schrödingerove jednadžbe, jer ukoliko znamo H, znamo u potpunosti dinamiku sustava. Saznavanje hamiltonijana sustava nije lagan posao te uključuje mnogo kompleksnih eksperimenata.

5.1.3. Mjerenje kvantnog sustava

Zatvoreni kvantni sustav evoluira prema Schrödingerovoj jednadžbi, međutim ukoliko u određenom trenutku želimo vidjeti što se događa unutar samog sustava, moramo obaviti mjerenje nad sustavom. Kada napravimo mjerenje kvantnoga sustava, taj kvantni sustav više nije zatvoren. Treći postulat kvantne mehanike govori o efektima mjerenja na stanje kvantnoga sustava [70][83].

Postulat 3: Kvantna mjerenja su opisana kolekcijom $\{M_m\}$, koja se sastoji od operatora mjerenja. Takvi operatori djeluju na stanje sustava koje se mjeri. Indeks m se odnosi na izmjereni ishod koji se može dogoditi prilikom mjerenja sustava [83]. Ako je kvantni sustav u stanju $|\psi\rangle$ neposredno prije samog mjerenja, tada je vjerojatnost da će rezultat mjerenja biti m jednaka

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle,$$

dok sustav nakon mjerenja ostaje u stanju

$$|\psi'\rangle = \frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}.$$

Operatori mjerenja zadovoljavaju jednadžbu potpunosti

$$\sum_m M_m^\dagger M_m = I$$

Suma vjerojatnosti svih mogućih rezultata mjerenja je jednaka

$$1 = \sum_m p(m) = \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle\,.$$

5.1.4. Složeni kvantni sustav

Složeni kvantni sustav je kvantni sustav koji se sastoji od dva ili više pojedinačnih kvantnih sustava. Četvrti postulat kvantne mehanike opisuje prostor stanja složenog kvantnoga sustava [70][83].

Postulat 4: Prostor stanja složenog kvantnoga sustava je tenzorski produkt prostora stanja komponenata složenog kvantnoga sustava. Na primjer, ukoliko imamo kvantne sustave označene brojevima od 1 do *n*, te ukoliko je kvantni sustav *i* pripremljen u stanju $|\psi_i\rangle$, onda je složeni sustav sastavljen od tih *n* kvantnih sustava u stanju $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$ [83].

5.2. Reverzibilno sklopovlje

U klasičnom računarstvu elementarni logički sklopovi poput *I*, *ILI*, *NE* realiziraju neku Booleovu funkciju $f : \boldsymbol{B}^k \rightarrow \boldsymbol{B}$, gdje je $\boldsymbol{B} = \{0, 1\}$, a *k* prirodni broj. Kombiniranjem više klasičnih logičkih sklopova dobivamo logičko sklopovlje koje realizira neku funkciju $\{0,1\}^n \rightarrow \{0,1\}^m$ gdje su *n* i *m* prirodni brojevi.

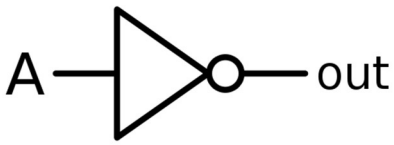
Logički sklop *L* je reverzibilan ukoliko za bilo koji izlaz *y* postoji jedinstveni ulaz *x* takav da je *L(x) = y*. Reverzibilni sklopovi imaju isti broj ulaznih i izlaznih bitova. Klasični sklopovi uglavnom nisu reverzibilni. Primjer jednog takvog sklopa je *ekskluzivni ILI* logički sklop.

Slika 1: Simbol ekskluzivnog ILI logičkog sklopa

Tablica 2: Tablica istinitosti za ekskluzivni ILI logički sklop

Ulaz		Izlaz
0	0	0
0	1	1
1	0	1
1	1	0

Dakle, na temelju vrijednosti izlaza iz *ekskluzivnog ILI* sklopa nije moguće na jednoznačan način odrediti koja je vrijednost bila na ulazu sklopa. Primjer reverzibilnog sklopovlja je logički sklop *NE*.



Slika 2: Simbol NE logičkog sklopa

Tablica 3: Tablica istinitosti za NE sklop

Ulaz	Izlaz
0	1
1	0

Dakle, na temelju vrijednosti izlaza iz *NE* sklopa moguće je na jednoznačan način odrediti koja je vrijednost bila na ulazu.

Svaki ireverzibilni logički sklop moguće je pretvoriti u reverzibilni sklop pomoću dodatnih bitova informacija na izlazu iz sklopovlja. Na taj način obogaćujemo izlaz informacijama koje su nam potrebne da zaključimo što je bilo na ulazu u sklop. Na primjer, reverzibilna verzija *ekskluzivnog ILI* sklopa prikazana je u tablici 4.

Tablica 4: Tablica istinitosti za reverzibilni ekskluzivni ILI sklop

Ulaz		Izlaz	
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Vidimo da na temelju vrijednosti svih izlaza možemo jednoznačno odrediti koje informacije su bile na ulazu u reverzibilni *ekskluzivni ILI* sklop. Treba napomenuti da ne vrijedi nužno tvrdnja da izjednačavanje broja ulaznih i izlaznih bitova sklopa ima za posljedicu reverzibilan sklop. Naime, sagledajmo tablicu istinitosti za potpuno zbrajalo koja je ireverzibilna.

Tablica 5: Tablica istinitosti za potpuno zbrajalo

Ulaz			Izlaz	
C_{ulaz}	x	y	C_{izlaz}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Ukoliko dodamo jedan dodatni bit (g_1) na izlaz potpunog zbrajala i dalje ne možemo jednoznačno odrediti ulaz na temelju izlaza iz potpunog zbrajala. Tek nakon dodavanja dodatnog bita na ulaz, te dodatna dva bita na izlaz, tablica istinitosti za potpuno zbrajalo postaje reverzibilna [27].

Tablica 6: Reverzibilna tablica istinitosti za potpuno zbrajalo

Ulaz				Izlaz			
a_1	C_{ulaz}	x	y	C_{izlaz}	S	g_1	g_2
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1
1	0	1	0	0	0	1	0
1	0	1	1	0	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	1

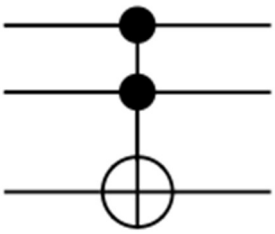
U klasičnom računarstvu *NAND* vrata su univerzalna [75], što znači da je bilo koje klasično logičko sklopovlje moguće implementirati pomoću samo *NAND* vrata.

Slika 3: Simbol NAND logičkih vrata

Tablica 7: Tablica istinitosti za NAND logička vrata

Ulaz		Izlaz
0	0	1
0	1	1
1	0	1
1	1	0

Reverzibilna *NAND* logička vrata se mogu realizirati pomoću reverzibilnih Toffolijevih vrata [46], što pak povlači da su Toffolijeva vrata ujedno i univerzalna za reverzibilno sklopovlje.

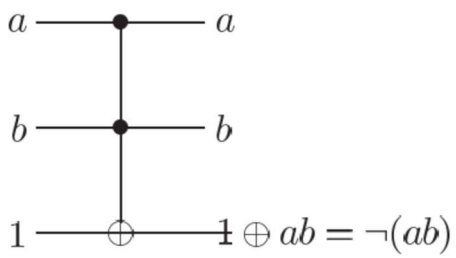


Slika 4: Simbol Toffolijevih vrata

Tablica 8: Tablica istinitosti Toffolijevih vrata

Ulaz			Izlaz		
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



Slika 5: Realizacija NAND vrata pomoću Toffolijevih vrata

5.3. Kvantni bit

Najjednostavniji kvantnomehanički sustav je *qubit*, odnosno kvantni bit. Kvantni bit je analogan klasičnom bitu i predstavlja jedinicu kojom se mjeri količina kvantne informacije. Kvantni bit ima dvodimenzionalni prostor stanja u kojem vektori $|0\rangle$ i $|1\rangle$ čine ortonormalnu bazu. Proizvoljni vektor u dvodimenzionalnom prostoru možemo napisati kao

$$|\psi\rangle = a|0\rangle + b|1\rangle \qquad (5.3.1.),$$

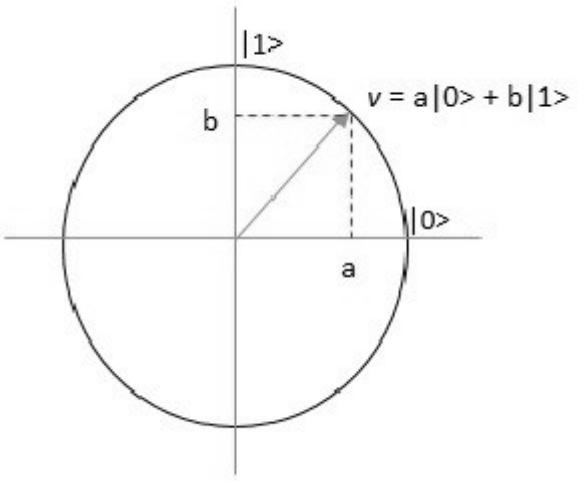
gdje su *a* i *b* kompleksni brojevi koji se nazivaju amplitudama. Ako želimo da je $|\psi\rangle$ jedinični vektor, odnosno da je skalarni produkt $\langle\psi|\psi\rangle$ jednak 1, tada mora vrijediti da je $|a|^2 + |b|^2 = 1$. Stanja $|0\rangle$ i $|1\rangle$ u kvantnom bitu su analogna stanjima 0 i 1 u klasičnom bitu. Kvantni bit se razlikuje od klasičnoga bita u tome što on može biti u superponiranom stanju ta dva stanja $|0\rangle$ i $|1\rangle$. U superponiranom stanju nije moguće sa sigurnošću reći nalazi li se kvantni bit u stanju $|0\rangle$ ili stanju $|1\rangle$, dok je kod klasičnoga bita ta informacija uvijek poznata.

Razlikujemo dva tipa kvantnih stanja: čista i mješovita. Čista kvantna stanja su ona stanja koja se mogu opisati preko jednog *ket* vektora, kao što je to prikazano u izrazu (5.3.1.). Mješovita stanja se ne mogu prikazati pomoću jednog *ket* vektora, već ona predstavljaju statističku kombinaciju čistih stanja. Mješovita stanja se prikazuju pomoću matrica gustoće

$$\rho = \sum_s p_s \langle\psi_s|\psi_s\rangle,$$

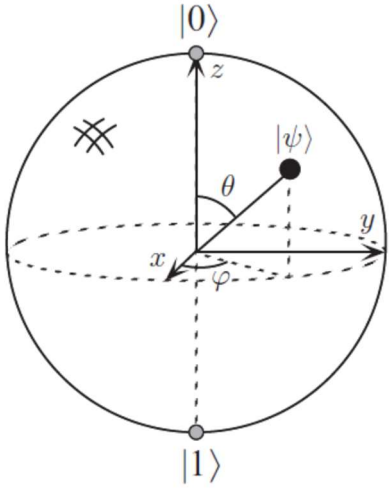
gdje su $|\psi_s\rangle$ čista stanja.

Vektor stanja kvantnoga sustava za jedan kvantni bit, čije amplitude u realnoj domeni možemo vizualizirati na brojevnoj kružnici.



Slika 6: Vizualizacija vektora stanja kvantnoga bita s realnim amplitudama

Ukoliko amplitude vektora stanja kvantnoga bita sadrže i imaginarne komponente, vektor stanja možemo vizualizirati na Blochovoj kugli



Slika 7: Vizualizacija vektora stanja kvantnoga bita s kompleksnim amplitudama

gdje je $|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$.

Točke koje se nalaze na površini Blochove kugle predstavljaju čista stanja, dok točke koje se nalaze u unutrašnjosti kugle predstavljaju mješovita stanja. U kvantnom računarstvu stanja se najčešće prikazuju u bazi $|0\rangle$ i $|1\rangle$, međutim to ne mora nužno biti tako. Baza može biti proizvoljna, ali bitno je da bazni vektori budu međusobno ortogonalni, kako bi na temelju rezultata mjerenja kvantnoga sustava mogli razlikovati u koje bazno stanje se sustav urušio. Dokaz kontradikcijom je u nastavku.

Pretpostavimo da imamo dva bazna stanja, $|\psi_1\rangle$ i $|\psi_2\rangle$, koja međusobno nisu ortonormalna, te da je moguće napraviti mjerenje pomoću kojeg se može jednoznačno utvrditi u koje se stanje sustav nakon mjerenja urušio. Ako je sustav u stanju $|\psi_1\rangle$, neka je vjerojatnost da će rezultat mjerenja biti 1 jednak $p_1 = 1$. Isto vrijedi i za $|\psi_2\rangle$, tj. ako je sustav u stanju $|\psi_2\rangle$, onda je vjerojatnost da će rezultat biti 2 jednaka $p_2 = 1$.

Definirajmo operatore mjerenja

$$E_i \equiv \sum_{j:f(j)=i} M_j^\dagger M_j.$$

Dakle,

$$p_1 = \langle\psi_1|E_1|\psi_1\rangle = 1,\ p_2 = \langle\psi_2|E_2|\psi_2\rangle = 1. \quad (5.3.2.)$$

S obzirom da mora vrijediti $\sum_i E_i = I$, slijedi da je $\sum_i\langle\psi_1|E_i|\psi_1\rangle = 1$, odnosno $\langle\psi_1|E_1|\psi_1\rangle = 1$, $\langle\psi_1|E_2|\psi_1\rangle = 0$, $\sqrt{E_2}|\psi_1\rangle = 0$. Pretpostavimo da dekomponiramo $|\psi_2\rangle = \alpha|\psi_1\rangle + \beta|\varphi\rangle$, gdje je $|\varphi\rangle$ ortonormalan sa $|\psi_1\rangle$. Zbog toga što je $|\alpha|^2 + |\beta|^2 = 1$, kao i zbog toga što $|\psi_1\rangle$ i $|\psi_2\rangle$ nisu ortonormalni, mora vrijediti $|\beta| < 1$. Možemo zaključiti da bi trebalo vrijediti $\sqrt{E_2}|\psi_2\rangle = \beta\sqrt{E_2}|\varphi\rangle$, što je kontradiktorno s (5.3.2.) jer

$$\langle\psi_2|E_2|\psi_2\rangle = |\beta|^2\langle\varphi|E_2|\varphi\rangle \leq |\beta|^2 < 1.$$

5.4. Kvantna prepletenost

1935. godine Einstein, Podolsky i Rosen su u svojem poznatom radu „*Can quantum-mechanical description of physical reality be considered complete?*“ opisali paradoks kojim

su htjeli pokazati da je teorija kvantne mehanike nepotpuna [29]. Taj paradoks je nazvan „EPR paradoks“. John Bell je 1964. godine osmislio misaoni eksperiment koji je omogućio da se eksperimentalno utvrdi je li kvantna mehanika odnosno „EPR paradoks“ doista način na koji priroda funkcionira ili pak postoje skrivene varijable, kako su u svojem radu tvrdili Einstein, Podolsky i Rosen. Kasnije je to imalo za posljedicu otkrivanje fenomena koji se zove kvantna prepletenost. Bell je svoje rezultate objavio u radu „*On the Einstein-Podolsky-Rosen Paradox*“ [5]. Kvantna prepletenost je fizički resurs kojeg dijele fizički razdvojeni kvantni sustavi [50], te se ona može mjeriti. Bellova stanja predstavljaju četiri specifična stanja kvantnomehaničkog sustava s dva maksimalno prepletena kvantna bita.

Bellova stanja su:

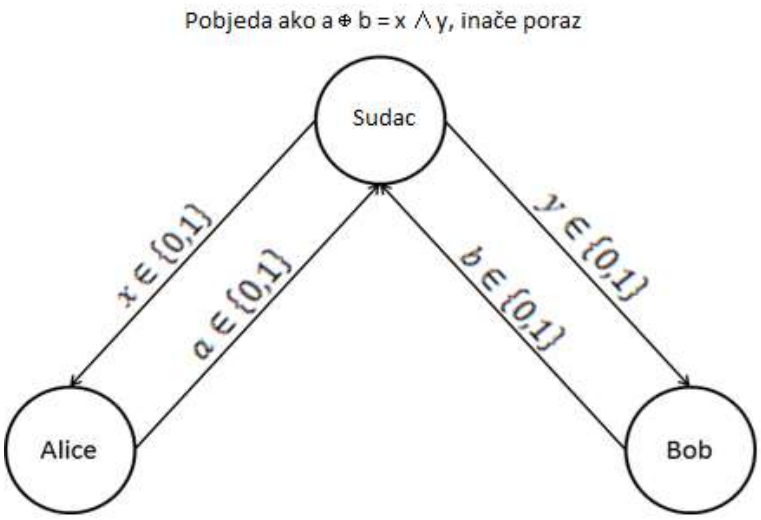
$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

$$|\beta_{01}\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$$

$$|\beta_{10}\rangle = \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$$

$$|\beta_{11}\rangle = \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$$

Mnogi igrači u kooperativnim igrama iz kvantne teorije igara koriste upravo kvantnu prepletenost kako bi stekli prednost u odnosu na klasičnu verziju iste igre. Jedna od najpoznatijih kvantnih igara je CHSH igra koja je dobila ime po njezinim autorima F. Clauser, M. A. Horne, A. Shimony, i R. A. Holt [16]. CHSH igru igraju dva igrača, Alice i Bob, koji su međusobno udaljeni i ne mogu nikako komunicirati u klasičnom smislu. Sudac daje Alici nasumičan bit *x*, a Bobu nasumičan bit *y*. Alice odgovara sucu s bitom *a,* a Bob s bitom *b*. Sudac gleda sve bitove *x,y,a,b* te proglašava jesu li Alice i Bob pobjednici ili gubitnici u partiji igre. Sudac odlučuje na temelju sljedećeg pravila — ako je istina da je $a \oplus b = x \wedge y$, Alice i Bob su pobijedili u partiji igre, u protivnom su izgubili. Simbol $\oplus$ označava XOR operaciju (zbrajanje modulo 2).



Slika 8: CHSH igra

Kada bi Alice i Bob igrali klasičnu verziju igre bez razmjene ikakvih informacija putem klasičnoga kanala, vjerojatnost da pobijede je $\frac{3}{4}$. Tu vjerojatnost mogu postići ukoliko igraju prema sljedećoj strategiji:

1. Bob uvijek odgovara s $b = 0$
2. Ako Alice od suca dobije $x = 0$, onda će ona odgovoriti s $a = 0$. Alice i Bob će u tom slučaju pobijediti s vjerojatnošću 1.
3. Ako Alice od suca dobije x = 1, onda se ona mora kockati sa svojim odgovorom, odnosno vjerojatnost da Alice i Bob pobijede u ovom slučaju je $\frac{1}{2}$.

Sveukupna vjerojatnost da Alice i Bob pobijede primjenjujući opisanu strategiju je

$$P_c = \frac{1}{2} * 1 + \frac{1}{2} * \frac{1}{2} = \frac{3}{4}.$$

Međutim, kada bi Alice i Bob imali na raspolaganju kvantnomehanički sustav s dva kvantna bita koji je prethodno inicijaliziran u Bellovom stanju $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, vjerojatnost za dobitak bi bila oko 0,8 ako primjenjuju sljedeću strategiju [2].

1. Ako Alice od suca dobije $x = 1$, ona na svoj kvantni bit primjenjuje rotaciju od $\frac{\pi}{8}$.

$$R_A = \begin{bmatrix} \cos\left(\frac{\pi}{8}\right) & -\sin\left(\frac{\pi}{8}\right) \\ \sin\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) \end{bmatrix}$$

2. Ako Alice dobije od suca $x = 0$, ne primjenjuje nikakvu operaciju na svoj kvantni bit.
3. Ako Bob dobije od suca $y = 1$, on na svoj kvantni bit primjenjuje rotaciju od $-\frac{\pi}{8}$.

$$R_B = \begin{bmatrix} \cos\left(\frac{\pi}{8}\right) & \sin\left(\frac{\pi}{8}\right) \\ -\sin\left(\frac{\pi}{8}\right) & \cos\left(\frac{\pi}{8}\right) \end{bmatrix}$$

4. Ako Bob dobije od suca $x = 0$, ne primjenjuje nikakvu operaciju na svoj kvantni bit.
5. Alice i Bob mjere svoje kvantne bitove, a vrijednosti koje dobivaju šalju kao odgovor sucu.

Ako Alice dobije od suca $x = 0$, a Bob $y = 0$, nijedno ne primjenjuju nikakve operacije nad kvantnim bitovima pa kvantni sustav ostaje u stanju $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, što znači da Alice i Bob pobjeđuju s vjerojatnošću 1.

Ako Alice dobije od suca $x = 0$, a Bob $y = 1$, Alice ne primjenjuje nikakve operacije na svoj kvantni bit, dok Bob na svoj kvantni bit primjenjuje operaciju R_B. Kvantni sustav je u stanju

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\left(|0\rangle\left(\cos\left(\frac{\pi}{8}\right)|0\rangle - \sin\left(\frac{\pi}{8}\right)|1\rangle\right)\right) + \left(|1\rangle\left(\sin\left(\frac{\pi}{8}\right)|0\rangle + \cos\left(\frac{\pi}{8}\right)|1\rangle\right)\right)\right)$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\cos\left(\frac{\pi}{8}\right)|00\rangle - \sin\left(\frac{\pi}{8}\right)|01\rangle + \sin\left(\frac{\pi}{8}\right)|10\rangle + \cos\left(\frac{\pi}{8}\right)|11\rangle\right).$$

Dakle, Alice i Bob u ovoj situaciji pobjeđuju s vjerojatnošću $\cos^2\left(\frac{\pi}{8}\right)$.

Ako Alice dobije od suca $x = 1$, a Bob $y = 0$, Alice na svoj kvantni bit primjenjuje operaciju R_A, dok Bob na svoj kvantni bit ne primjenjuje nikakvu operaciju. Kvantni sustav je u stanju

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\left(\left(\cos\left(\frac{\pi}{8}\right)|0\rangle + \sin\left(\frac{\pi}{8}\right)|1\rangle\right)|0\rangle\right) + \left(\left(-\sin\left(\frac{\pi}{8}\right)|0\rangle + \cos\left(\frac{\pi}{8}\right)|1\rangle\right)|1\rangle\right)\right)$$

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\cos\left(\frac{\pi}{8}\right)|00\rangle - \sin\left(\frac{\pi}{8}\right)|01\rangle + \sin\left(\frac{\pi}{8}\right)|10\rangle + \cos\left(\frac{\pi}{8}\right)|11\rangle\right).$$

Alice i Bob u ovoj situaciji pobjeđuju s vjerojatnošću $\cos^2\left(\frac{\pi}{8}\right)$.

Ako Alice dobije od suca $x = 1$, a Bob $y = 1$, Alice na svoj *qubit* primjenjuje R_A operaciju, dok Bob na svoj *qubit* primjenjuje operaciju R_B. Kvantni sustav je u stanju

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\left(\left(\cos\left(\frac{\pi}{8}\right)|0\rangle + \sin\left(\frac{\pi}{8}\right)|1\rangle\right)\left(\cos\left(\frac{\pi}{8}\right)|0\rangle - \sin\left(\frac{\pi}{8}\right)|1\rangle\right)\right) + \left(\left(\left(-\sin\left(\frac{\pi}{8}\right)|0\rangle + \cos\left(\frac{\pi}{8}\right)|1\rangle\right)\right)\left(\sin\left(\frac{\pi}{8}\right)|0\rangle + \cos\left(\frac{\pi}{8}\right)|1\rangle\right)\right)\right)$$

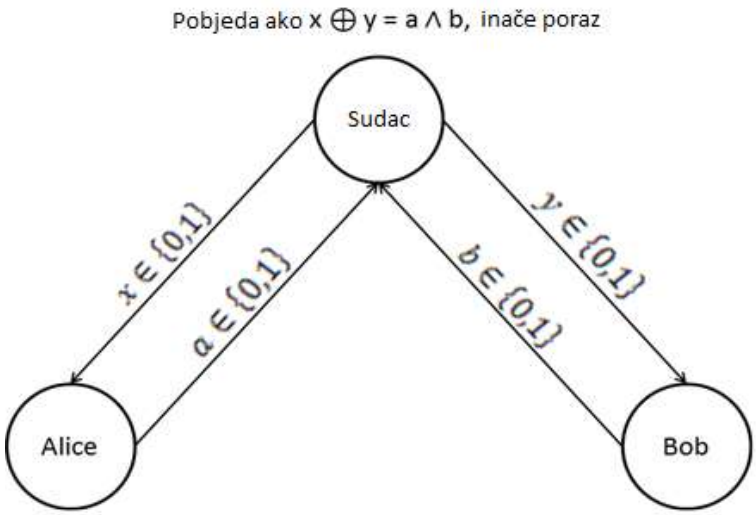
$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\left(\cos^2\left(\frac{\pi}{8}\right) - \sin^2\left(\frac{\pi}{8}\right)\right)|00\rangle - 2\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)|01\rangle + 2\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)|10\rangle + \left(\cos^2\left(\frac{\pi}{8}\right) - \sin^2\left(\frac{\pi}{8}\right)\right)|11\rangle\right).$$

S obzirom da je $\left(\cos^2\left(\frac{\pi}{8}\right) - \sin^2\left(\frac{\pi}{8}\right)\right) = \cos\left(\frac{\pi}{4}\right) = \sin\left(\frac{\pi}{4}\right) = 2\sin\left(\frac{\pi}{8}\right)\cos\left(\frac{\pi}{8}\right)$, sve amplitude baznih stanja imaju istu apsolutnu vrijednost, dakle Alice i Bob u ovoj situaciji pobjeđuju sa vjerojatnošću od $\frac{1}{2}$.

Ukupna vjerojatnost da Alice i Bob pobijede primjenjujući opisanu kvantnu strategiju je

$$P_q = \frac{1}{4} * 1 + \frac{1}{4} * \cos^2\left(\frac{\pi}{8}\right) + \frac{1}{4} * \cos^2\left(\frac{\pi}{8}\right) + \frac{1}{4} * \frac{1}{2} \approx 0{,}8.$$

Još izrazitija prednost kvantne strategije nad klasičnom vidljiva je u igri sličnoj CHSH igri u kojoj Alice i Bob dobivaju igru ako vrijedi $x \oplus y = a \wedge b$ [7].



Slika 9: Igra temeljena na CHSH igri

U klasičnoj verziji igre očigledno je da se Alice i Bob moraju uvijek kockati sa svojim odgovorima, odnosno vjerojatnost da Alice i Bob pobijede je $\frac{1}{2}$.

$$P_c = \frac{1}{2}.$$

Međutim, ako Alice i Bob odigraju kvantnu verziju igre, vjerojatnost pobjede je znatno veća ako koriste sljedeću strategiju.

1. Alice i Bob dijele kvantnomehanički sustav s dva kvantna bita koji je inicijaliziran u Bellovom stanju $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. Alice uzima prvi kvantni bit, dok Bob uzima drugi kvantni bit.
2. Ako Alice od suca dobije x = 1, ona na svoj kvantni bit primjenjuje Hadamardov operator $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$. Ako Alice od suca dobije x = 0, ona ne primjenjuje nikakvu operaciju na svoj kvantni bit.
3. Ako Bob od suca dobije y = 1, on na svoj kvantni bit primjenjuje Hadamardov operator $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$. Ako Bob od suca dobije y = 0, on ne primjenjuje nikakvu operaciju na svoj kvantni bit.

4. Alice i Bob mjere svoje kvantne bitove, a vrijednosti koje dobivaju šalju kao odgovor sucu.

Ako Alice od suca dobije x = 0, a Bob y = 0, oni ne primjenjuju operacije na svoje kvantne bitove, stoga kvantni sustav ostaje u stanju $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$. Alice i Bob pobjeđuju s vjerojatnošću 1.

Ako Alice od suca dobije x = 0, a Bob y = 1, Alice ne primjenjuje nikakvu operaciju na svoj kvantni bit, dok Bob na svoj primjenjuje Hadamardov operator. Kvantni sustav je u stanju

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

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|01\rangle\right) - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle\right)$$

$$|\psi\rangle = \frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle.$$

Alice i Bob u ovom slučaju pobjeđuju s vjerojatnošću $\frac{1}{4}$.

Ako Alice od suca dobije x = 1, a Bob y = 0, Alice primjenjuje Hadamardov operator na svoj kvantni bit, dok Bob na svoj ne primjenjuje nikakvu operaciju. Kvantni sustav je u stanju

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

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) - \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|10\rangle\right)$$

$$|\psi\rangle = \frac{1}{2}|01\rangle + \frac{1}{2}|11\rangle - \frac{1}{2}|00\rangle + \frac{1}{2}|10\rangle.$$

Alice i Bob u ovom slučaju pobjeđuju s vjerojatnošću $\frac{1}{4}$.

Ako Alice od suca dobije x = 1, a Bob y = 1, Alice i Bob će na svoje kvantne bitove primijeniti Hadamardov operator. Kvantni sustav je u stanju

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

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(\frac{1}{2}|00\rangle - \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle\right) - \frac{1}{\sqrt{2}}\left(\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle - \frac{1}{2}|10\rangle - \frac{1}{2}|11\rangle\right)$$

$$|\psi\rangle = -\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle.$$

Alice i Bob u ovom slučaju pobjeđuju s vjerojatnošću 1.

Sveukupna vjerojatnost da Alice i Bob pobijede u kvantnoj verziji igre primjenom opisane strategije je

$$P_q = \frac{1}{4} * 1 + \frac{1}{4} * \frac{1}{4} + \frac{1}{4} * \frac{1}{4} + \frac{1}{4} * 1 = 0{,}625.$$

5.5. Kvantna dekoherencija

Kvantna dekoherencija je gubitak kvantne informacije koja je vezana uz probabilističke amplitude baznih stanja kvantnomehaničkog sustava koji je u superpoziciji. Kvantna dekoherencija se javlja kada je kvantni sustav u interakciji s okolinom, što predstavlja veliki izazov u fizičkoj realizaciji kvantnoga računala [28].

Postoje dva krajnja slučaja interakcije kvantnomehaničkog sustava sa svojom okolinom. Prvi je slučaj kada okolina u potpunosti apsorbira kvantnomehanički sustav, odnosno kada se on „stopi“ s okolinom [44]. Drugi je idealizirani slučaj u kojem okolina nema nikakav utjecaj na kvantnomehanički sustav, dok kvantnomehanički sustav ima utjecaj na okolinu.

Za primjer uzmimo kvantnomehanički sustav koji je u stanju

$$|\psi\rangle = \sum_i |i\rangle\langle i|\psi\rangle,$$

gdje je $|i\rangle$ skup ortonormalnih vektora. Stanje okoline možemo označiti s $|\epsilon\rangle$. Ukoliko sagledavamo okolinu i kvantni sustav kao cjelinu, onda je ukupno stanje ta dva kvantna sustava određen tenzorskim produktom stanja ta dva sustava, odnosno

$$|prije\rangle = \sum_i |i\rangle|\epsilon\rangle\langle i|\psi\rangle. \quad (5.5.1.)$$

Ukoliko okolina u potpunosti apsorbira kvantni sustav, $|i\rangle|\epsilon\rangle$ evoluira u $|\epsilon_i\rangle$ te je novo stanje složenog kvantnoga sustava

$$|poslije\rangle = \sum_i |\epsilon_i\rangle\langle i|\psi\rangle,$$

pri čemu pravilo ortonormalnosti i dalje mora ostati zadovoljeno, odnosno $\langle i|j\rangle = \delta_{ij}$ i $\langle\epsilon_i|\epsilon_j\rangle = \delta_{ij}$.

Ukoliko okolina nema utjecaj na kvantni sustav, produkt $|i\rangle|\epsilon\rangle$ iz (5.5.1.) evoluira u $|i\rangle|\epsilon\rangle = |i\rangle|\epsilon_i\rangle$, pri čemu je razvidno da sustav ima utjecaj na okolinu, dok okolina nema utjecaj na sustav. Sustav je u stanju

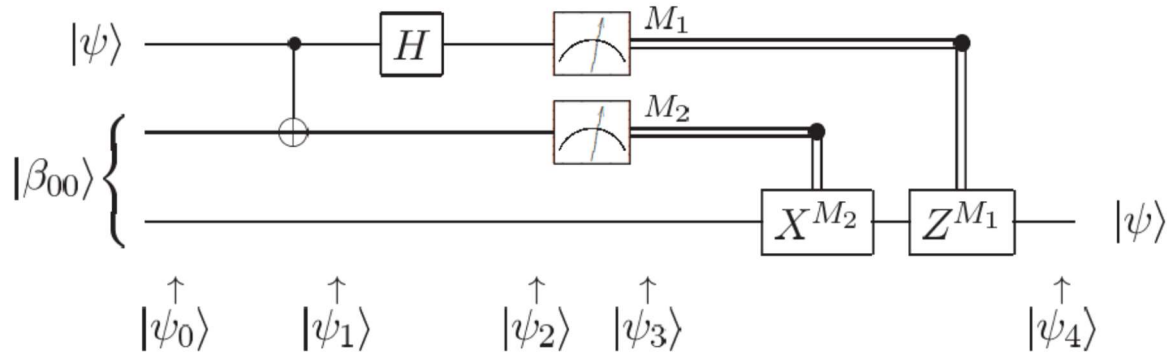
$$|poslije\rangle = \sum_i |i\rangle|\epsilon_i\rangle\langle i|\psi\rangle,$$

gdje mora biti zadovoljeno $\langle i, \epsilon_i|j, \epsilon_j\rangle = \langle i|j\rangle\langle\epsilon_i|\epsilon_j\rangle = \delta_{ij}\langle\epsilon_i|\epsilon_j\rangle = \delta_{ij}$. U praksi se najčešće pojavljuje slučaj koji je zapravo kombinacija spomenuta dva krajnja slučaja.

5.6. Kvantna teleportacija

Kvantna teleportacija je postupak pomoću kojeg je moguće prenositi kvantno stanje u prostoru [109]. Taj koncept bit će objašnjen kroz sljedeći primjer.

Alice i Bob su fizički udaljeni jedan od drugog te dijele kvantni sustav koji je inicijaliziran u Bellovom stanju $|\beta_{00}\rangle$. Cilj je omogućiti Alice da prenese bilo koje kvantno stanje jednog kvantnoga bita Bobu. Alice i Bob na raspolaganju imaju klasični informacijski kanal, međutim potrebno je napomenuti da Alice ne zna koje kvantno stanje treba poslati Bobu, a čak i kada bi ga znala, klasični informacijski kanal „pati“ od pogreške u zaokruživanju koja bi se pojavila ukoliko bi se kvantno stanje pokušalo prenijeti klasičnim kanalom. Da bi Alice uspjela poslati Bobu proizvoljno kvantno stanje, oni moraju koristiti kvantni informacijski kanal i primijeniti postupak pod nazivom kvantna teleportacija.



Slika 10: Kvantno sklopovlje za kvantnu teleportaciju

Stanje$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ na slici 10 predstavlja stanje kvantnoga bita kojeg Alice mora prenijeti Bobu, dok je *EPR* par kojeg dijele Alice i Bob inicijalno u Bellovom stanju $|\beta_{00}\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Kvantni sustav sa slike 10 inicijalno je u stanju

$$|\psi_0\rangle = |\psi\rangle|\beta_{00}\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)].$$

Prvi korak je da Alice radi interakciju kvantnoga bita koji je u stanju $|\psi\rangle$ sa svojim kvantnim bitom *EPR* para upotrebom *CNOT* kvantnih vrata, stavljajući cijeli sustav u stanje

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)].$$

Nakon toga Alice primjenjuje Hadamardova vrata na prvi kvantni bit, te stavlja cijeli kvantni sustav u stanje

$$|\psi_2\rangle = \frac{1}{2}[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)]$$

$$= \frac{1}{2}[|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)]. \quad (5.6.1.)$$

Iz kvantnoga stanja prikazanog u (5.6.1.) vidimo da je Bobov kvantni bit „preuzeo“ amplitude stanja kvantnoga bita čije stanje je Alice poslala, međutim Bob mora napraviti korektivne radnje ovisno o informaciji koju će dobiti od Alice, koja mu mora poslati rezultate

mjerenja prva dva kvantna bita sa slike 10. Dakle, Bobov kvantni bit će nakon mjerenja biti u jednom od sljedećih stanja, ovisno o Aliceinim mjerenjima:

$$00 \rightarrow |\psi_3(00)\rangle \equiv [\alpha|0\rangle + \beta|1\rangle]$$

$$01 \rightarrow |\psi_3(01)\rangle \equiv [\alpha|1\rangle + \beta|0\rangle]$$

$$10 \rightarrow |\psi_3(10)\rangle \equiv [\alpha|0\rangle - \beta|1\rangle]$$

$$11 \rightarrow |\psi_3(00)\rangle \equiv [\alpha|1\rangle - \beta|0\rangle].$$

Ovisno o informaciji dobivenoj od Alice, Bob popravlja stanje u svojem kvantnom bitu primjenjujući jedna od četiriju kvantnih vrata.

$$00 \rightarrow I$$

$$01 \rightarrow X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$10 \rightarrow Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$11 \rightarrow XZ$$

U konačnici je Bobov kvantni bit u stanju $|\psi\rangle$, što je i bio prvobitni cilj.

5.7. „No cloning“ teorem

Teorem o nemogućnosti kloniranja kvantnoga stanja govori o tome da nepoznato kvantno stanje nije moguće klonirati odnosno kopirati. Dokaz je u nastavku [83].

Pretpostavimo da imamo kvantni uređaj koji ima dva kvantna bita, kvantni bit A i kvantni bit B. Kvantni bit A sadrži proizvoljno kvantno stanje $|\psi\rangle$ koje treba kopirati u kvantno stanje kvantnoga bita B. Također, pretpostavimo da je kvantni bit B pripremljen u nekom inicijalnom standardnom stanju $|s\rangle$. Dakle, sustav je u početnom stanju $|\psi\rangle|s\rangle$. Sada zamislimo da postoji neki unitarni operator U koji će kopirati stanje iz kvantnoga bita A u stanje kvantnoga bita B.

$$|\psi\rangle|s\rangle \rightarrow U(|\psi\rangle|s\rangle) = |\psi\rangle|\psi\rangle.$$

Pretpostavimo da unitarni operator dobro radi za dva čista proizvoljna stanja $|\psi\rangle$ i $|\varphi\rangle$.

$$U(|\psi\rangle|s\rangle) = |\psi\rangle|\psi\rangle$$

$$U(|\varphi\rangle|s\rangle) = |\varphi\rangle|\varphi\rangle.$$

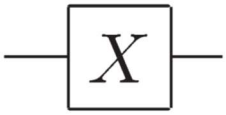
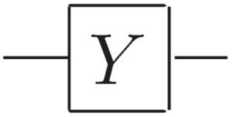
Skalarni produkt tih dviju jednadžbi je $\langle\psi|\varphi\rangle = (\langle\psi|\varphi\rangle)^2$, što povlači da je $|\psi\rangle = |\varphi\rangle$ ili da su $|\psi\rangle$ i $|\varphi\rangle$ ortogonalni. Možemo zaključiti da nije moguće napraviti kvantno kloniranje ukoliko stanja koja se kloniraju nisu međusobno ortogonalna.

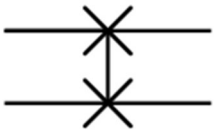
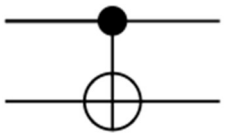
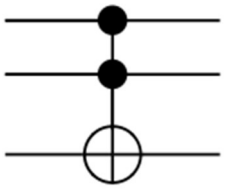
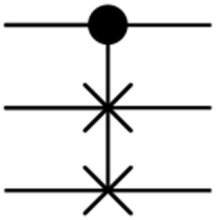
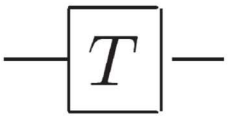
5.8. Kvantna vrata

Kvantna vrata su analogna klasičnim logičkim vratima te predstavljaju bazične kvantne sklopove kojima su realizirane određene unitarne operacije nad manjim brojem kvantnih bitova. Pomoću jednostavnijih kvantnih vrata realiziraju se složeniji kvantni sklopovi, koji pak primjenjuju složenije unitarne operacije na veći broj kvantnih bitova.

Neke od najčešće spominjanih kvantnih vrata u literaturi su

Tablica 9: Najpoznatija kvantna vrata

Vrata	Operator	Simbol
Hadamardova vrata	$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	
Paulijeva-X vrata	$X = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$	
Paulijeva-Y vrata	$Y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$	

Paulijeva-Z vrata	$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	
Phase shift vrata	$R_\theta = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$	
Swap vrata	$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	
CNOT vrata	$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	
Toffolijeva vrata	$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	
Fredkinova vrata	$F = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	
$\frac{\pi}{8}$ vrata	$\frac{\pi}{8} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	

5.8.1. Kontrolirana vrata

Kontrolirana vrata se primjenjuju ako je uvjet zadovoljen. Uvjet se testira preko kontrolnih kvantnih bitova, a operacija se izvršava na odredišnim kvantnim bitovima. Na primjer, uzmimo da je

$$U = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}$$

proizvoljna unitarna operacija. Kontrolna kvantna vrata *C(U)* koja bi primijenila unitarnu operaciju *U* na odredišni drugi kvantni bit u ovisnosti o kontrolnom prvom kvantnom bitu bi izgledala ovako:

$$C(U) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & x_{00} & x_{01} \\ 0 & 0 & x_{10} & x_{11} \end{bmatrix}$$

Primjer kontroliranih vrata su CNOT, Toffolijeva i Fredkinova vrata.

5.8.2. Univerzalni skup kvantnih vrata

S obzirom da se Toffolijeva vrata mogu realizirati na kvantnom računala i da su univerzalna za klasično sklopovlje (realiziraju NAND vrata), a nisu univerzalna za kvantno sklopovlje (npr. ne mogu realizirati superpoziciju dva bazična stanja), možemo zaključiti da je klasično sklopovlje podskup kvantnog sklopovlja [46][75][83].

Jedan od univerzalnih skupova za kvantno računalo pomoću kojeg je moguće realizirati bilo koje kvantno sklopovlje je skup: Hadamardova vrata, CNOT vrata, $\frac{\pi}{8}$ vrata [83]. Da bismo došli do tog zaključka treba sagledati nekoliko sljedećih točaka.

5.8.2.1. Skup dvorazinskih unitarnih vrata je univerzalan

Pomoću sljedećeg dokaza pokazat ćemo da je bilo koju unitarnu operaciju moguće konstruirati pomoću dvorazinskih kvantnih vrata. Dvorazinska kvantna vrata su vrata koja netrivijalno djeluju na podprostor prostora u kojem se nalazi vektor stanja kvantnoga sustava, a taj podprostor je razapet dvama baznim vektorima.

Uzmimo da je U proizvoljna 3×3 unitarna matrica

$$U = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & j \end{bmatrix}.$$

Cilj je pronaći takve matrice da vrijedi produkt

$$U_3 U_2 U_1 U = I,$$

odnosno

$$U = U_1^\dagger U_2^\dagger U_3^\dagger.$$

U_1, U_2, U_3 su dvorazinske unitarne matrice, te su njihove inverzne matrice $U_1^\dagger, U_2^\dagger, U_3^\dagger$ također dvorazinske unitarne matrice.

U nastavku je opisan postupak pomoću kojeg je moguće izvesti matrice U_1, U_2, U_3. Ako je $b = 0$, tada vrijedi da je U_1

$$U_1 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Ako je $b \neq 0$, tada je U_1

$$U_1 \equiv \begin{bmatrix} \frac{a^*}{\sqrt{|a|^2 + |b|^2}} & \frac{b^*}{\sqrt{|a|^2 + |b|^2}} & 0 \\ \frac{b}{\sqrt{|a|^2 + |b|^2}} & \frac{-a}{\sqrt{|a|^2 + |b|^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Kada pomnožimo U sa U_1 dobivamo

$$U_1 U = \begin{bmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{bmatrix}.$$

Istu logiku primjenjujemo i za c'. Ako je c' = 0, tada je

$$U_2 \equiv \begin{bmatrix} a'^* & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Inače, ako je $c' \neq 0$, tada je

$$U_2 \equiv \begin{bmatrix} \frac{a'^*}{\sqrt{|a'|^2 + |c'|^2}} & 0 & \frac{c'^*}{\sqrt{|a'|^2 + |c'|^2}} \\ 0 & 1 & 0 \\ \frac{c'}{\sqrt{|a'|^2 + |c'|^2}} & 0 & \frac{-a'}{\sqrt{|a'|^2 + |c'|^2}} \end{bmatrix},$$

te vrijedi

$$U_2 U_1 U = \begin{bmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{bmatrix}.$$

S obzirom da su U, U_1, U_2 unitarne matrice, mora vrijediti da je matrica $U_2 U_1 U$ također unitarna, odnosno da je d''=g''=0, jer prvi red od matrice $U_2 U_1 U$ mora imati normu 1. Konačno, U_3 matrica izgleda ovako:

$$U_3 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & e''^* & f''^* \\ 0 & h''^* & j''^* \end{bmatrix}.$$

Lako se može provjeriti vrijedi li $U_3U_2U_1U = I$, odnosno $U = U_1^\dagger U_2^\dagger U_3^\dagger$.

U općem slučaju, kada želimo dekomponirati matricu dimenzije d na ovakav način, dekompozicija će izgledati ovako:

$$U = V_1 \dots V_k,$$

gdje je $k \leq (d-1) + (d-2) + \cdots + 1 = d(d-1)/2$.

5.8.2.2. Skup kvantnih vrata nad jednim kvantnim bitom i CNOT vrata je univerzalan

U ovom dijelu cilj nam je pokazati da je pomoću kvantnih vrata nad jednim kvantnim bitom i *CNOT* kvantnih vrata moguće konstruirati bilo koju dvorazinsku unitarnu matricu. Na taj način ćemo zaključiti da je taj skup kvantnih vrata univerzalan skup za kvantno sklopovlje.

Pretpostavimo da je U dvorazinska unitarna matrica nad kvantnim računalom koje ima n kvantnih bitova, te da U djeluje netrivijalno nad podprostorom koji je razapet vektorima $|s\rangle$ i $|t\rangle$, gdje je $s = s_1 \dots s_n$, i $t = t_1 \dots t_n$ binarna reprezentacija tih vektora. Također, uzmimo da je U' 2×2 netrivijalna unitarna podmatrica matrice U.

Sada ćemo koristiti tehniku zvanu „sivi kod“, kojoj je cilj krenuti od binarne reprezentacije vektora $|s\rangle$ te završiti u binarnoj reprezentaciji vektora $|t\rangle$, s tim da se u svakom prijelazu binarni broj smije razlikovati od prethodnog samo u jednom bitu.

Npr. recimo da je $s = 101001$ i da je $t = 110011$.

Tada sivi kod za $s \to t$ izgleda ovako:

$$\begin{matrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{matrix}$$

Recimo da su $g_1, g_2 \dots g_n$ elementi sivog koda koji povezuju s i t. $g_1 = s$, a $g_n = t$. Ideja je pronaći takav slijed kvantnih vrata G koja mijenjaju stanja $|g_1\rangle \to |g_2\rangle \to \cdots \to |g_{m-1}\rangle$, a zatim primijeniti kontrolirana U' vrata, s ciljnim kvantnim bitom na onom dijelu gdje se razlikuju g_m i g_{m-1}, a zatim poništiti utjecaj slijeda kvantnih vrata G.

Konkretna implementacija se radi na sljedeći način. Prvi korak je da se zamijene stanja $|g_1\rangle$ i $|g_2\rangle$, a to radimo tako da uvodimo kontrolirana *NE* vrata čiji je odredišni kvantni bit na onoj poziciji gdje se $|g_1\rangle$ i $|g_2\rangle$ u binarnoj reprezentaciji razlikuju, dok su ostali kvantni bitovi kontrolni. Drugi korak je da uvodimo nova kontrolirana *NE* vrata za $|g_2\rangle$ i $|g_3\rangle$, i tako sve dok ne dođemo do stanja $|g_{m-1}\rangle$. Sada primjenjujemo kontrolirana U' vrata na onaj kvantni bit na čijoj se poziciji binarne reprezentacije $|g_{m-1}\rangle$ i $|g_m\rangle$ razlikuju, dok su ostali kvantni bitovi kontrolni. Na kraju treba poništiti utjecaj uvedenih kontroliranih *NE* vrata u prijašnjim koracima tako da se ponovno primjenjuje isti slijed kontroliranih *NE* vrata, ali obrnutim redoslijedom.

Recimo da želimo prema opisanom postupku dekomponirati dvorazinsku unitarnu matricu

$$U = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{bmatrix}$$

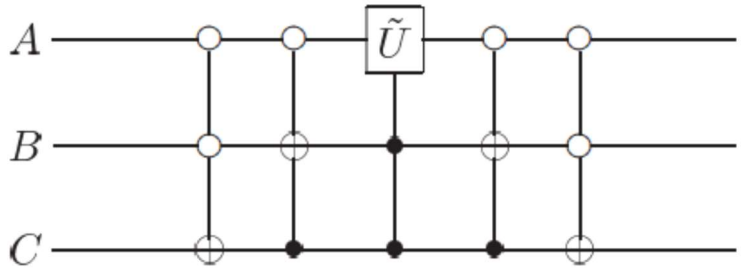
te da je

$$U' = \begin{bmatrix} a & c \\ b & d \end{bmatrix}.$$

Dakle, U netrivijalno djeluje na stanja $|000\rangle$ i $|111\rangle$, a sivi kod za te prijelaze između stanja izgleda ovako:

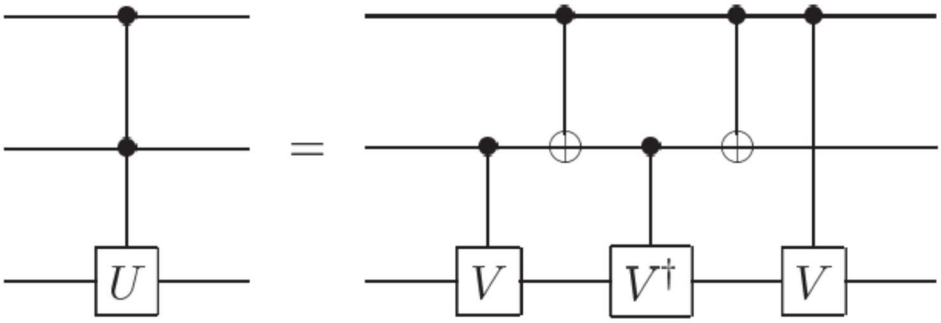
A	B	C
0	0	0
0	0	1
0	1	1
1	1	1

Prema opisanom postupku, konačni slijed kvantnih vrata koji implementiraju unitarnu matricu U izgleda ovako:



Slika 11: Kvantno sklopovlje za dvorazinsku unitarnu matricu U

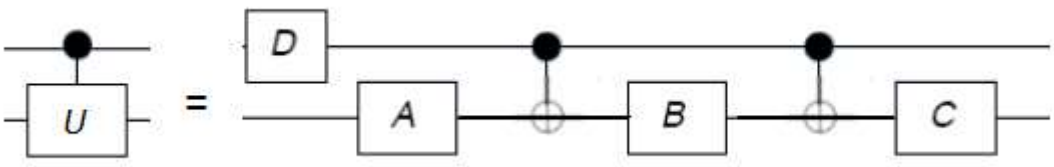
Isto tako, treba napomenuti da je bilo koja kontrolirana vrata moguće dekomponirati na niz jednostavnijih kvantnih vrata na način koji je prikazan na sljedećoj slici



Slika 12: Dekompozicija kontroliranih vrata

gdje je V korijen matrice U. Dakle možemo zaključiti da kontrolirana vrata nad jednim kvantnim bitom u kombinaciji sa *CNOT* vratima čine univerzalni skup vrata za kvantno sklopovlje.

Svaka kontrolirana kvantna vrata koja imaju jedan odredišni i jedan kontrolni kvantni bit se mogu dekomponirati na niz kvantnih vrata koja djeluju nad jednim kvantnim bitom [17]



Slika 13: Dekompozicija kontroliranih vrata s jednim odredišnim i jednim kontrolnim kvantnim bitom

gdje se U može napisati kao

$$U = \begin{bmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{bmatrix} \begin{bmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2} \\ -\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \begin{bmatrix} e^{-i\beta/2} & 0 \\ 0 & e^{i\beta/2} \end{bmatrix}$$

dok se matrice A, B, C i D mogu napisati kao

$$A = \begin{bmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{bmatrix} \begin{bmatrix} \cos\frac{\theta}{4} & \sin\frac{\theta}{4} \\ -\sin\frac{\theta}{4} & \cos\frac{\theta}{4} \end{bmatrix}$$

$$B = \begin{bmatrix} \cos -\frac{\theta}{4} & \sin -\frac{\theta}{4} \\ -\sin -\frac{\theta}{4} & \cos -\frac{\theta}{4} \end{bmatrix} \begin{bmatrix} e^{i(\alpha+\beta)/4} & 0 \\ 0 & e^{-i(\alpha+\beta)/4} \end{bmatrix}$$

$$C = \begin{bmatrix} e^{i(\alpha-\beta)/4} & 0 \\ 0 & e^{-i(\alpha-\beta)/4} \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\delta} \end{bmatrix}.$$

U poglavlju 5.9.2.1 smo vidjeli da bilo koju unitarnu matricu možemo implementirati pomoću dvorazinskih matrica, dok smo u ovom poglavlju pokazali da bilo koju dvorazinsku matricu možemo dekomponirati u skup kvantnih vrata koja djeluju nad jednim kvantnim bitom i *CNOT* vrata. Dakle, zaključak je da su *CNOT* vrata u kombinaciji s vratima koja djeluju nad jednim kvantnim bitom univerzalni skup kvantnih vrata za kvantno sklopovlje.

5.8.2.3. Skup Hadamardovih i $\frac{\pi}{8}$ vrata je univerzalan

U ovom dijelu ćemo pokazati da je bilo koju unitarnu operaciju nad jednim kvantnim bitom moguće dekomponirati na niz Hadamardovih (*H*) i $\frac{\pi}{8}$ (*T*) kvantnih vrata [83].

T vrata su zapravo rotacija od $\frac{\pi}{4}$ radijana oko *z* osi u Blochovoj kugli, dok je slijed vrata *THT* rotacija od $\frac{\pi}{4}$ radijana oko *x* osi u Blochovoj kugli. Komponiranjem tih dviju operacija dobivamo

$$e^{-i\frac{\pi}{8}Z}e^{-i\frac{\pi}{8}X} = \left[\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}Z\right]\left[\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}X\right]$$
$$= \cos^2\frac{\pi}{8}I - i\left[\cos\frac{\pi}{8}(X+Z) + \sin\frac{\pi}{8}Y\right]\sin\frac{\pi}{8},$$

što je u biti rotacija Blochove kugle za kut θ koji je definiran kao $\cos\frac{\theta}{2} \equiv \cos^2\frac{\pi}{8}$ preko $\vec{n} = (\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8})$, gdje je $\vec{n}$ odgovarajući jedinični vektor. Znači, upotrebom Hadamardovih i $\frac{\pi}{8}$ vrata možemo konstruirati operatora rotacije $R_n(\theta)$. Također se može pokazati da je θ iracionalni višekratnik od 2π [83]. Sada moramo pokazati da uzastopnom upotrebom rotacije $R_n(\theta)$ možemo aproksimirati rotaciju $R_n(\alpha)$ s proizvoljnom točnošću, gdje je α proizvoljan kut. Da bismo to vidjeli, uzmimo da je $\delta > 0$ željena točnost, te da je N cijeli broj koji je veći od $\frac{2\pi}{\delta}$. Definirajmo θ_k tako da je $\theta_k \epsilon [0, 2\pi)$ i $\theta_k = (k\theta) mod\ 2\pi$. Tada postoje različiti j i k u domeni $1, \dots, N$ takvi da vrijedi $|\theta_k - \theta_j| \leq \frac{2\pi}{N} < \delta$. Bez gubitka općenitosti, možemo pretpostaviti da je $k > j$, stoga vrijedi $|\theta_{k-j}| < \delta$. S obzirom da je $j \neq k$ i θ iracionalni višekratnik od 2π, vrijedi da je $\theta_{k-j} \neq 0$. Slijedi da slijed $\theta_{l(k-j)}$ popunjava interval $[0, 2\pi)$ kako l varira, tako da susjedni članovi nisu udaljeni više od δ, što je definirana točnost. Na kraju možemo zaključiti da za svaki $\epsilon > 0$ postoji s takav da je

$$E(R_n(\alpha), R_n(\theta)^s) < \frac{\epsilon}{3}.$$

Sada smo u poziciji u kojoj možemo pokazati da svaka unitarna operacija nad jednim kvantnim bitom može biti aproksimirana do željene točnosti upotrebom Hadamardovih i $\frac{\pi}{8}$ vrata. Vrijedi da je

$$HR_n(\alpha)H = R_m(\alpha)$$

gdje je m jedinični vektor u smjeru $(\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8})$, dakle slijedi da je

$$E(R_m(\alpha), R_m(\theta)^s) < \frac{\epsilon}{3}.$$

S obzirom da se svaka unitarna operacija U nad jednim kvantnim bitom može napisati kao

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta)$$

gdje su R_z i R_y rotacijske matrice oko z, odnosno y osi za zadani kut (Z$-Y$ dekompozicija) [83], vrijedi

$$U = R_n(\beta) R_m(\gamma) R_n(\delta),$$

odnosno

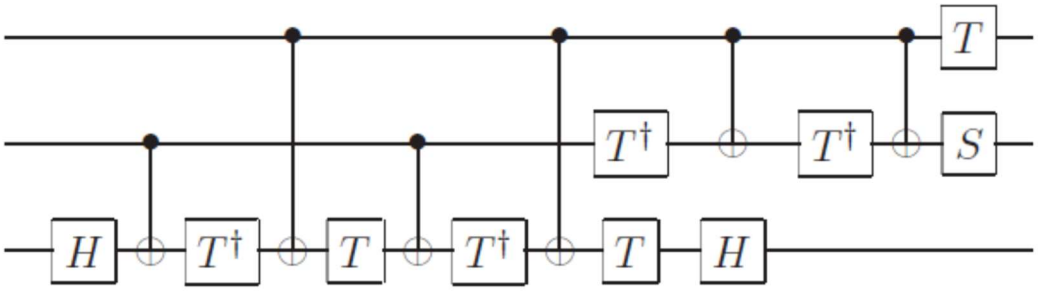
$$E(U, R_n(\theta)^{s_1} H R_n(\theta)^{s_2} H R_n(\theta)^{s_3}) < \frac{\epsilon}{3},$$

gdje su s_1, s_2 i s_3 odgovarajući pozitivni cijeli brojevi. Dakle, pomoću Hadamardovih (H) i $\frac{\pi}{8}$ (T) kvantnih vrata možemo aproksimirati bilo koju unitarnu operaciju U nad jednim kvantnim bitom do proizvoljne točnosti.

Na temelju dokaza iz poglavlja 5.8.2.1., 5.8.2.2. i 5.8.2.3. možemo zaključiti da pomoću skupa Hadamardovih, CNOT i $\frac{\pi}{8}$ kvantnih vrata možemo aproksimirati proizvoljnu unitarnu operaciju koja djeluje na proizvoljni broj kvantnih bitova.

5.9. Kvantno sklopovlje

Kvantno sklopovlje je slijed kvantnih vrata kojim je realizirana neka unitarna operacija nad skupom kvantnih bitova (kvantni registar) [112].



Slika 14: Kvantno sklopovlje za Toffolijeva vrata

Kada želimo usporediti trošak kvantnih sklopova koji izvršavaju neku unitarnu operaciju nad skupom kvantnih bitova, to možemo napraviti mjereći duljinu kvantnog sklopovlja, broj dodatnih kvantnih bitova, kvantni trošak, trošak tranzistora, trošak najbližeg susjeda ili dubinu sklopovlja [111]. Duljina kvantnog sklopovlja je broj kvantnih vrata od kojih je sastavljeno kvantno sklopovlje. Često duljina kvantnog sklopovlja nije reprezentativan pokazatelj samog troška sklopovlja jer se svaki kvantni sklop može svesti na jedna kvantna vrata koja djeluju na cijeli kvantni registar (2. postulat kvantne mehanike – poglavlje 5.1.2.). Drugi pokazatelj troška je broj dodatnih kvantnih bitova koje je potrebno pridodati samom kvantnom sklopovlju kako bi se mogla realizirati određena kvantna vrata od kojih se isti sastoji. Naime, što je takvih dodatnih kvantnih bitova manje, sklopovlje je jednostavnije pa je time i trošak manji. Najčešće upotrebljavana mjera za trošak kvantnog sklopovlja je kvantni trošak, odnosno broj elementarnih kvantnih vrata od kojih se sklopovlje sastoji. Elementarnim kvantnim vratima se smatraju kvantna vrata koja djeluju nad jednim ili dva kvantna bita. U literaturi se spominju i druge manje korištene mjere, poput troška tranzistora kod kojeg izračun troška ovisi o broju kontrolnih kvantnih bitova vrata od kojeg se kvantno sklopovlje sastoji. Isto tako, u literaturi se spominje i mjera trošak najbližeg susjeda, koji se odnosi na trošak koji nastaje prilikom interakcije između kvantnih bitova u fizičkoj realizaciji. Postoji i mjera dubina sklopovlja, koja mjeri broj potrebnih koraka da se izvedu sva vrata u kvantnom sklopovlju, s obzirom da se neka vrata mogu izvoditi paralelno ako ne djeluju na zajedničke podskupove kvantnih bitova u jednom trenutku.

5.10. Kvantni algoritmi

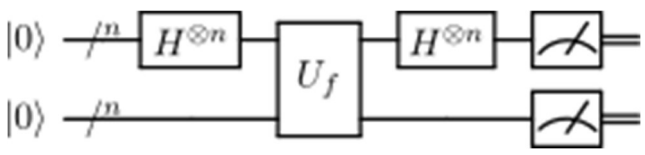
Algoritmi kvantnoga računala nisu intuitivni čovjeku zato što se moraju uzeti u obzir kvantnomehanički efekti poput superpozicije i prepletenosti [83]. Postoji nekoliko tehnika izgradnje kvantnih algoritama. Tehnika jačanja amplitude baznog stanja se svodi na to da kvantni algoritam kroz svoje iteracije pojačava amplitudu onog baznog stanja koje predstavlja rješenje problema [70][83][86]. Algoritam prolazi kroz iteracije sve dok vjerojatnost baznog stanja koje predstavlja rješenje ne bude 1. Na kraju se vrši mjerenje kvantnoga sustava, nakon čega se dobiva rješenje. Druga popularna tehnika je upotreba kvantne Fourierove transformacije pomoću koje se postižu superpolinomna ubrzanja u odnosu na klasično računalo [70][83] [86].

Nekoliko algoritama za kvantno računalo ukazuju na to da bi kvantno računalo moglo imati značajno više računske snage nego klasično računalo. Jedan od značajnijih kvantnih algoritama je Groverov algoritam koji rješava problem pronalaska elementa u nesortiranoj listi pomoću kvantnoga računala [38]. Najbolji primjer snage kvantnoga računala je Shorov algoritam koji rješava problem kanonske faktorizacije cijelog broja u eksponencijalno bržem vremenu nego je to trenutno moguće na klasičnom računalu [87]. U nastavku će biti opisani neki poznatiji algoritmi za kvantna računala.

5.10.1. Simonov algoritam

Simonov algoritam je kvantni algoritam koji rješava Simonov problem. Simonov problem se svodi na pronalaženje periode p za funkciju $f: \{0,1\}^n \rightarrow \{0,1\}^n$ za koju vrijedi [81]:

1. za svaki x postoji $y \neq x$ za kojeg vrijedi $f(x) = f(y)$
2. f je periodička funkcija, što znači da postoji $p \in \{0,1\}^n$ za kojeg vrijedi $f(x \oplus \mathrm{p}) = f(x), \forall x \in \{0,1\}^n$, gdje je $\oplus$ binarni operator zbrajanja modulo 2.



Slika 15: Kvantno sklopovlje za Simonov algoritam [62]

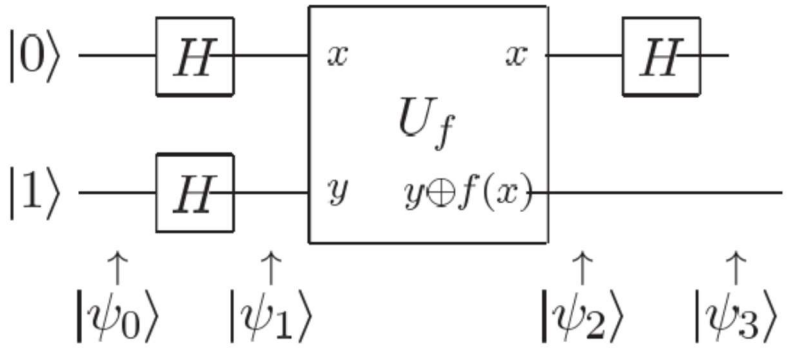
Početno stanje sustava je $|0\rangle^n|0\rangle^n$, nakon primjene Hadamardovog operatora na prvih n kvantnih bitova sustav je u stanju $\sum_x|x\rangle|0\rangle$. Operator U_f se u literaturi naziva prorok, a njegova primjena dovodi sustav u stanje $\sum_x|x\rangle|f(x)\rangle$, ponovna primjena Hadamardovog operatora na prvih n kvantnih bitova sustav dovodi u stanje $\sum_y\sum_x(-1)^{x\cdot y}|y\rangle|f(x)\rangle$ gdje je $\cdot$ binarni operator množenja modulo 2. Stanje prvog kvantnoga registra (prvih n kvantnih bitova) je

$$\sum_y((-1)^{x\cdot y}+(-1)^{(x\oplus \mathrm{p})\cdot y})\,|y\rangle=\sum_y((-1)^{x\cdot y}+(-1)^{x\cdot y}(-1)^{p\cdot y})\,|y\rangle$$

Za vrijednost y koja će se iščitati iz prvog kvantnoga registra mora vrijediti $p\cdot y=0$, u protivnom će amplituda biti jednaka 0. Nakon dovoljnog broja ponavljanja možemo dobiti $n-1$ linearno nezavisnih funkcija za koje vrijedi $p\cdot y=0$, te pomoću Gaussove eliminacije možemo izračunati p. Može se pokazati da je već nakon $2n$ ponavljanja velika vjerojatnost da ćemo dobiti $n-1$ linearno nezavisnih funkcija, dakle ukupna složenost Simonovog algoritma je $O(n)$, dok najbrži algoritam za klasična računala ima složenost $\Omega(2^{\frac{n}{2}})$ [62].

5.10.2. Deutschov algoritam

Deutschov algoritam rješava ne toliko bitan problem u računarstvu, međutim na dobar način pokazuje računalnu snagu kvantnoga računala naspram klasičnoga računala. Pretpostavimo da imamo funkciju $f:\{0,1\}\rightarrow\{0,1\}$. Vidimo da su moguća četiri preslikavanja za takvu funkciju. Deutschov algoritam odgovara da li je funkcija f bijekcija ili nije. To se može riješiti na način da se dobije vrijednost $f(0)\ XOR\ f(1)$, ako je ta vrijednost 1 onda je funkcija f bijekcija, u suprotnom funkcija f nije bijekcija [22].



Slika 16: Kvantno sklopovlje za Deutschov algoritam

Početno stanje za Deutschov algoritam je $|\psi_0\rangle = |01\rangle$, potom na svaki kvantni bit djeluje Hadamardov operator, dakle novo stanje sustava je $|\psi_1\rangle = \left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]$. Operator U_f djeluje na stanje $|x\rangle\frac{(|0\rangle-|1\rangle)}{\sqrt{2}}$ kao $(-1)^{f(x)}|x\rangle\frac{(|0\rangle-|1\rangle)}{\sqrt{2}}$. Nakon primjene operatora U_f na $|\psi_1\rangle$ dobivamo

$$|\psi_2\rangle = \begin{cases} \pm\left[\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \; ako\; f(0) = f(1) \\ \pm\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right]\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \; ako\; f(0) \neq f(1) \end{cases}$$

Primjenom Hadamardovog operatora na prvi kvantni bit dobivamo stanje

$$|\psi_3\rangle = \begin{cases} \pm|0\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \; ako\; f(0) = f(1) \\ \pm|1\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right] \; ako\; f(0) \neq f(1) \end{cases}$$

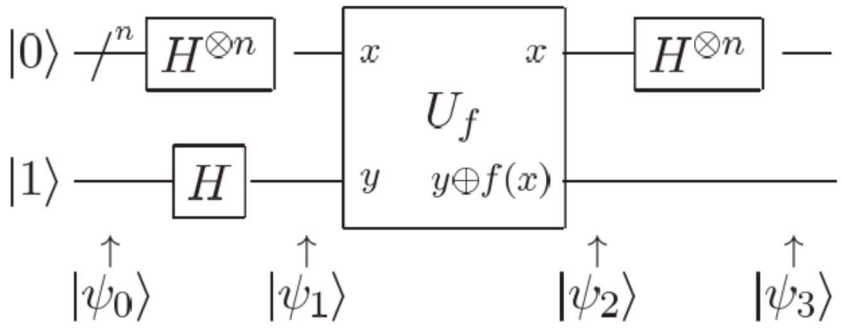
odnosno

$$|\psi_3\rangle = \pm|f(0) \oplus f(1)\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right].$$

Iščitavajući vrijednost prvog kvantnoga bita dobivamo 1 ukoliko je $f(0) = f(1)$, odnosno 0 ukoliko je $f(0) \neq f(1)$. Dakle, možemo zaključiti da je u ovom kvantnom algoritmu potrebna samo jedna evaluacija funkcije $f(x)$, dok je klasičnom računalu potrebno minimalno dvije evaluacije iste funkcije kako bi došlo do zaključka je li funkcija bijekcija ili nije.

5.10.3. Deutsch - Jozsa algoritam

Pretpostavimo da imamo funkciju $f: \{0,1\}^n \rightarrow \{0,1\}$. Deutsch-Jozsa algoritam odgovara na pitanje je li funkcija f balansirana ili konstantna [21]. Funkcija f je balansirana ako pola vrijednosti iz domene preslikava u vrijednost 0 iz kodomene, te ako pola vrijednosti iz domene preslikava u vrijednost 1 iz kodomene. Funkcija f je konstantna ako sve vrijednosti iz domene preslikava u jednu vrijednost iz kodomene.



Slika 17: Kvantno sklopovlje za Deutsch-Jozsa algoritam

Početno stanje za Deutsch-Josza algoritam je $|\psi_0\rangle = |0\rangle^n|1\rangle$. U prvom koraku algoritma primjenjuje se Hadamardov operator na prvih *n* kvantnih bitova, sustav je u stanju

$$|\psi_1\rangle = \sum_{x \in \{0,1\}^n} \frac{|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right].$$

Potom se primjenjuje U_f operatora na stanje sustava, sustav je u stanju

$$|\psi_2\rangle = \sum_x \frac{(-1)^{f(x)}|x\rangle}{\sqrt{2^n}} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right].$$

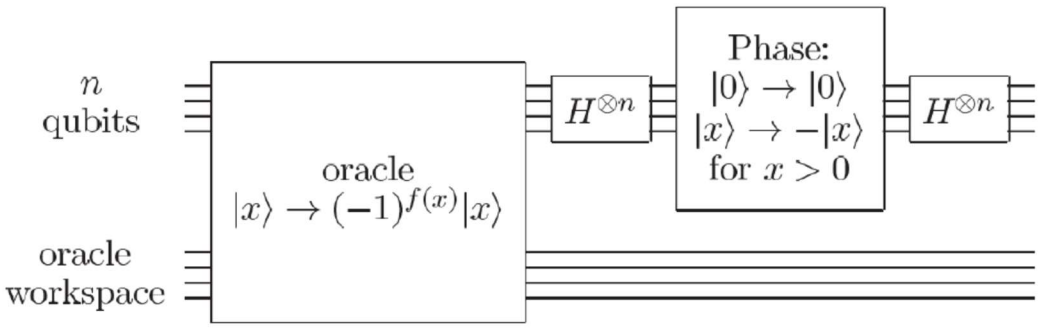
U zadnjem koraku se primjenjuje Hadamardov operator na prvih *n* kvantnih bitova. Sustav je u stanju

$$|\psi_3\rangle = \sum_y \sum_x \frac{(-1)^{x\cdot y + f(x)}|y\rangle}{2^n} \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right].$$

O tome je li f balansirana ili ne zaključujemo prema rezultatu čitanja prvih *n* kvantnih bitova. Ukoliko je rezultat jednak 0, radi se o konstantnoj funkciji, u suprotnom se radi o balansiranoj funkciji.

5.10.4. Groverov algoritam

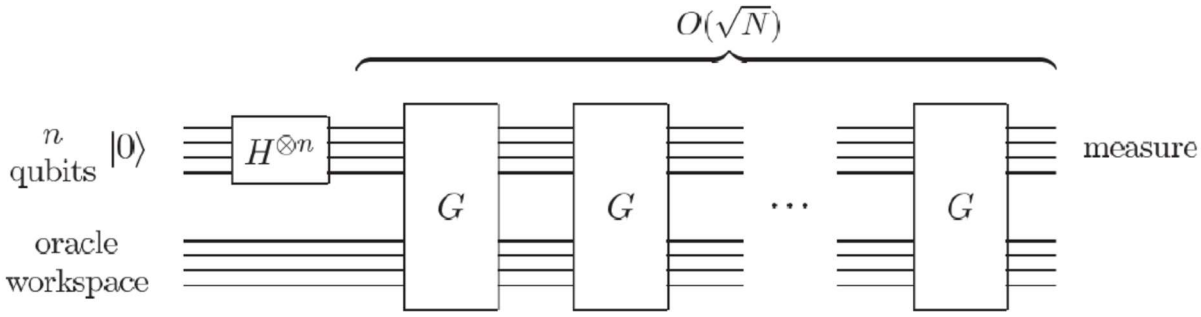
Groverov algoritam je kvantni algoritam koji pronalazi traženi element u nesortiranoj listi u vremenu $O(\sqrt{N})$, gdje je *N* broj elemenata u listi. Isto tako, dokazano je da je složenost od $O(\sqrt{N})$ optimalna za kvantna računala za problem pretraživanja nesortirane liste [115]. Prosječna složenost algoritma za klasična računala je $O\left(\frac{N}{2}\right)$. Groverov algoritam kroz svoje iteracije jača amplitudu onog baznog stanja koje predstavlja poziciju traženog elementa u listi.



Slika 18: Kvantno sklopovlje za jednu iteraciju Groverovog algoritma

Iteracija Groverovog algoritma koristi proroka (engl. *oracle*), koji identificira poziciju traženog elementa spremljenog u prvom kvantnom registru od *n* kvantnih bitova. Proroka

možemo shvatiti kao crnu kutiju koja ima mogućnost prepoznati rješenja problema. Prorok koristi funkciju $f(x)$ koja vraća 1 ukoliko stanje *x* predstavlja poziciju traženog elementa, u protivnom funkcija vraća vrijednost 0. Pomoću funkcije $f(x)$ prorok mijenja predznak amplitude pozicije (baznog stanja) na kojoj se nalazi traženi element. Kada prorok promijeni predznak pozicije traženog elementa, Hadamardova vrata, Phase shift vrata i opet Hadamardova vrata na prvih *n* kvantnih bitova jačaju amplitudu pozicije na kojoj se nalazi traženi element za $O\left(\frac{2}{\sqrt{N}}\right)$. U literaturi se ta tri kvantna operatora nazivaju i Groverov difuzijski operator, te zajedno daju unitarni operator $H^{\otimes n}(2|0\rangle\langle 0|-I)H^{\otimes n} = 2|\psi\rangle\langle\psi| - I$. Nakon $O\left(\sqrt{N}\right)$ iteracija, amplituda pozicije traženog elementa je 1, te se registar može izmjeriti.

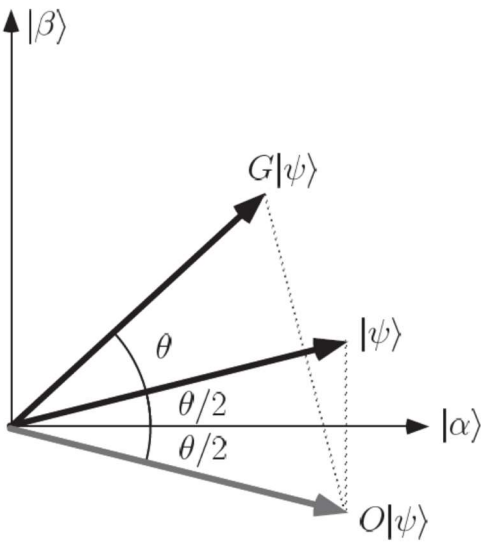


Slika 19: Kvantno sklopovlje za Groverov algoritam

Inicijalno stanje prvog kvantnoga registra u Groverovom algoritmu je $|\psi_0\rangle = |0\rangle$. Hadamardov operator postavlja registar u uniformnu superpoziciju svih mogućih baznih stanja

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle.$$

Grafički prikaz jedne iteracije Groverovog algoritma je prikazan na sljedećoj slici.



Slika 20: Grafički prikaz iteracije Groverovog algoritma

$|\beta\rangle$ predstavlja stanje koje je pozicija traženog elementa, dok $|\alpha\rangle$ predstavlja stanje koje nije rješenje problema. Groverov algoritam postupno pojačava amplitudu stanja $|\beta\rangle$ i slabi amplitudu stanja $|\alpha\rangle$. Hadamardov operator na početku Groverovog algoritma postavlja sustav u stanje $|\psi\rangle$. Prorok reflektira stanje $|\psi\rangle$ preko stanja $|\alpha\rangle$, dok Groverov difuzijski operator reflektira stanje $O|\psi\rangle$ preko stanja $|\psi\rangle$, odnosno pojačava amplitudu stanja $|\beta\rangle$ za $\frac{2}{\sqrt{N}}$. S obzirom da je ciljana amplituda stanja $|\beta\rangle$ 1, odnosno $\sin\left(\frac{\pi}{2}\right)$, potrebno je $\sqrt{N} * \frac{\pi}{4}$ iteracija Groverovog algoritma, što u konačnici daje ukupnu složenost od $O\left(\sqrt{N}\right)$. Može se pokazati da je, ukoliko se element koji tražimo u nesortiranoj listi pojavljuje nepoznati broj puta *t,* složenost takve verzije Groverovog algoritma $O(\sqrt{\frac{N}{t}})$. Takvu verziju Groverovog algoritma opisali su Michel Boyer, Gilles Brassard, Peter Høyer i Alain Tapp u njihovom radu „*Tight bounds on quantum searching*“ [10] .

Koraci Groverovog algoritma za nepoznati broj rješenja:

1. Neka je *T* nesortirana lista, a *x* tražena vrijednost. Inicijaliziraj *m* = 1 i λ = 6/5. (Bilo koja vrijednost između 1 i 4/3 je valjana za λ)
2. Slučajno izaberi *j* kao cijeli broj manji od m.

3. Inicijaliziraj kvantni registar veličine $n = \log_2 N$ u stanje $|\psi_0\rangle = |0\rangle$, nakon toga upotrijebi Hadamardov operator nad svakim kvantnim bitom. Kvantni registar je u stanju $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$.
4. Primjeni *j* iteracija Groverovog algoritma.
5. Iščitaj vrijednost iz registra — neka je *i* rezultat.
6. Ako je *T[i] = x*, problem je riješen te završi, inače idi na 7.
7. Postavi *m* na $min(\lambda m, \sqrt{N})$, te idi na korak 2.

5.10.5. Kvantni algoritam za pronalaženje maksimalne klike u neusmjerenom grafu

Groverov algoritam je imao veliki utjecaj na razvoj kvantnih algoritama, pa su mnogi autori predložili algoritme za različite probleme koji se baziraju na Groverovom algoritmu. Tako su, primjerice, Carlile Lavor, Leo Liberti i Nelson Maculan predložili kvantni algoritam za rješavanje problema molekularne geometrijske udaljenosti [71]. Paulo Mateus i Yasser Omar predložili su kvantni algoritam za najbliže poklapanje uzorka [76] . Mihai Udrescu, Lucian Prodan i Mircea Vladutiu predložili su metodologiju za izvršavanje genetskih algoritama na kvantnom računalu [103]. Riccardo Franco predložio je model ljudskog pamćenja [32]. Svi spomenuti algoritmi i modeli temelje se na Groverovom algoritmu.

Groverov algoritam se može iskoristiti i za rješavanje problema pronalaženja maksimalne klike u neusmjerenom grafu [9]. Maksimalna klika u neusmjerenom grafu *G=(V,E)* je najveći podskup skupa vrhova grafa, gdje su svi elementi/vrhovi tog podskupa međusobno povezani [26].

Kvantni algoritam funkcionira tako da sve podskupove skupa vrhova grafa „spremi“ u kvantni registar s jednakom vjerojatnosnom amplitudom. U grafu postoji $2^{|V|}$ takvih podskupova, stoga je $|V|$ broj potrebnih kvantnih bitova u kvantnom registru. Algoritam može postići takvo stanje primjenom Hadamardovog operatora na svaki kvantni bit u registru koji se inicijalno postavlja u stanje $|\psi_0\rangle = |0\rangle$. Dakle, kvantni registar će imati $2^{|V|}$ mogućih baznih stanja te će se na vrijednost koja je dobivena mjerenjem kvantnoga registra gledati kao na binarni broj. Ukoliko se radi o binarnoj znamenci 1 unutar binarnog broja, to znači da je taj vrh grafa uključen u podskup koji čini rezultat, u protivnom vrh nije uključen u podskup koji predstavlja rezultat.

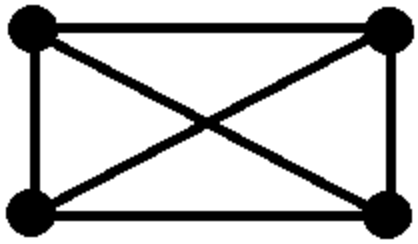
Obrada se izvodi tako da se koristi Groverov algoritam za pronalaženje klike čiji je kardinalni broj veći od nekog broja k koji je inicijalno postavljen na vrijednost 1. Kada algoritam pronađe takav podskup koji je klika zadanog grafa, a veličina te klike je veća od l, on pohranjuje veličinu te klike u varijablu k, a sam podskup u vektor $|r\rangle$ te se ponovno izvodi. Inicijalna vrijednost vektora $|r\rangle$ je $|0\rangle$. Algoritam završava tek kada veličina rezultirajuće klike više nije veća od vrijednosti u varijabli k ili kada dohvaćeni podskup više nije klika zadanog grafa. U konačnici, algoritam vraća vektor $|r\rangle$ te je, ukoliko je taj vektor drugačiji od $|0\rangle$, pronađena jedna od maksimalnih kliki zadanog grafa.

Dakle, koraci algoritma su:

1. Inicijaliziraj varijablu $k = 1$ i vektor $|r\rangle = |0\rangle$.
2. Definiraj funkciju *f(x)* za proroka koji će biti upotrijebljen unutar Groverovog algoritma. Funkcija *f(x)* vraća 1 ako bazno stanje x reprezentira kliku grafa te ukoliko je veličina klike veća od vrijednosti koja je pohranjena u varijabli k, inače funkcija vraća vrijednost 0.
3. Inicijaliziraj kvantni registar veličine $n = \log_2 N$ u stanje $|\psi_0\rangle = |0\rangle$. N je kardinalni broj partitativnog skupa nad skupom vrhova grafa V.
4. Upotrebom Hadamardovih vrata nad svakim kvantnim bitom postavi stanje sustava u $|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$.
5. Upotrijebi Groverov algoritam za nepoznati broj rješenja.
6. Izmjeri kvantni registar. Ako je rezultat mjerenja klika grafa, te je veličina te klike veća od vrijednosti u varijabli k, tada spremi taj rezultat u vektor $|r\rangle$, a veličinu klike u varijablu k, i vrati se na korak 2, u protivnom prijeđi na korak 7.
7. Vrati vektor $|r\rangle$.

Algoritmu je potrebno $|V|$ kvantnih bitova da bi se uspjela napraviti superpozicija svih podskupova skupa vrhova grafa V. Algoritmu je također potrebno prostora za pohranu trenutno pronađene maksimalne klike i njezine veličine. Za trenutnu kliku je potrebno $|V|$ klasičnih bitova, odnosno $\log_2 |V|$ za njezinu veličinu. Sveukupna memorijska složenost ovog algoritma je $O(|V|)$. Treba naglasiti da korak 6 ne utječe značajno na memorijsku kompleksnost algoritma zato što možemo odrediti veličinu skupa, te je li skup klika grafa u polinomnoj ovisnosti o $|V|$, čak i na klasičnim računalima.

Da bismo sagledali vremensku složenost algoritma analizirajmo sljedeći graf



Slika 21: Graf G u kojem je ω(G) = |V|

Maksimalna klika grafa uključuje sve vrhove tog grafa, dakle, veličina maksimalne klike u tom grafu iznosi *ω(G)*=4. U proizvoljnom grafu *G=(V,E)* postoji najmanje $2^{\omega(G)} - (\omega(G)+1)$ klika. S obzirom da je logika algoritma da iterativno povećava veličinu tražene klike, maksimalni broj iteracija algoritma je također jednak *ω(G)*. Iteracije algoritma u najgorem slučaju su:

1. iteracija — algoritam pronalazi kliku grafa veličine 2. U općem slučaju, kompleksnost ove iteracije iznosi $O\left(\sqrt{\frac{2^{|V|}}{\binom{\omega(G)}{2}+..+\binom{\omega(G)}{\omega(G)}}}\right)$, odnosno $O\left(\sqrt{\frac{2^{|V|}}{2^{\omega(G)}-(\omega(G)+1)}}\right)$. Za graf na slici 20, kompleksnost ove iteracije je $O\left(\sqrt{\frac{2^{|V|}}{11}}\right)$.

2. iteracija — algoritam pronalazi kliku grafa veličine 3. U općem slučaju, kompleksnost ove iteracije iznosi $O\left(\sqrt{\frac{2^{|V|}}{\binom{\omega(G)}{3}+..+\binom{\omega(G)}{\omega(G)}}}\right)$, odnosno $O\left(\sqrt{\frac{2^{|V|}}{2^{\omega(G)}-(\omega(G)+1)-\binom{\omega(G)}{2}}}\right)$. Za graf na slici 20, kompleksnost ove iteracije je $O\left(\sqrt{\frac{2^{|V|}}{4}}\right)$.

ω(G)-1. iteracija - algoritam pronalazi kliku grafa veličine *ω(G)*. U općem slučaju, kompleksnost ove iteracije iznosi $O\left(\sqrt{\frac{2^{|V|}}{\binom{\omega(G)}{\omega(G)}}}\right)$, odnosno $O\left(\sqrt{2^{|V|}}\right)$.

ω(G). iteracija - algoritam nije uspio pronaći kliku grafa koja je veća od *ω(G)*. U općem slučaju, kompleksnost ove iteracije iznosi $O(\sqrt{2^{|V|}})$.

Vidimo da je najgori slučaj kada graf ima maksimalnu kliku čija je veličina jednaka kardinalnom broju skupa vrhova grafa V, te kada se iz iteracije u iteraciju vrijednost veličine trenutno pronađene klike uvećava za 1. U najgorem slučaju zadnja iteracija ima najveću kompleksnost iz razloga što postoji samo jedna klika čija je veličina jednaka kardinalnom broju skupa vrhova grafa V. Složenost zadnje iteracije je $O(\sqrt{2^{|V|}})$. Kako *ω(G)* maksimalno može biti $|V|$, ukupna složenost cijelog algoritma je $O(|V|\sqrt{2^{|V|}})$.

Vjerojatnost pojave najgoreg mogućeg scenarija je

$$P_{wc} \geq \frac{\binom{\omega(G)}{2}}{\binom{\omega(G)}{2}+..+\binom{\omega(G)}{\omega(G)}} *..* \frac{\binom{\omega(G)}{\omega(G)}}{\binom{\omega(G)}{\omega(G)}} = \sum_{i=2}^{\omega(G)} \binom{\omega(G)}{i} \left(\sum_{j=i}^{\omega(G)} \binom{\omega(G)}{j} \right)^{-1}.$$

Za graf koji je prikazan na 21. slici, vjerojatnost pojave najgoreg mogućeg scenarija je $P_{wc} \geq \frac{6}{11} * \frac{4}{5} * 1 = \frac{24}{55}$. Isto tako, najbolji slučaj je kada zadani graf nema klike. Tada će algoritam završiti nakon samo jedne iteracije čija je složenost $O(\sqrt{2^{|V|}})$. Vjerojatnost pojave najboljeg mogućeg scenarija je

$$P_{bc} \geq \frac{1}{\binom{\omega(G)}{2}+..+\binom{\omega(G)}{\omega(G)}} = \left(\sum_{i=2}^{\omega(G} \binom{\omega(G)}{i} \right)^{-1}$$

Za graf koji je prikazan na 21. slici, vjerojatnost pojave najboljeg mogućeg scenarija je $P_{bc} \geq \frac{1}{6+4+1} = \frac{1}{11}$. Isto tako treba naglasiti da korak 6 ne utječe značajno na vremensku kompleksnost algoritma zato što možemo odrediti veličinu skupa, te je li skup klika grafa u polinomnoj vremenskoj ovisnosti o $|V|$, čak i na klasičnim računalima.

5.10.6. Shorov algoritam

Shorov algoritam rješava problem faktorizacije cijelih brojeva u polinomnom vremenu $O((\log N)^3)$, gdje je N broj koji se faktorizira [96]. Algoritam se temelji se na kvantnoj Fourierovoj transformaciji koja je definirana sa

$$|j\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{\frac{2\pi i}{2^n}} |k\rangle$$

gdje su $|j\rangle$ bazna stanja $0 \leq j \leq 2^n - 1$.

Kvantna Fourierova transformacija na proizvoljnom stanju kvantnoga sustava se može napisati kao

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left[\sum_{j=0}^{2^n-1} e^{\frac{2\pi i j k}{2^n}} x_j \right] |k\rangle = \sum_{k=0}^{2^n-1} y_k |k\rangle.$$

Ideja Shorovog algoritma je pronalazak periode *r* funkcije $f(x) = a^x \bmod N$, gdje je *a* proizvoljan relativno prosti broj s *N*, te je $a < N$. *N* je cijeli broj koji se faktorizira.

Koraci kvantnoga algoritma za pronalazak perioda *r* su:

1. Potrebno je izabrati broj Q koji je potencija broja 2, te je u intervalu $N^2 \leq Q \leq 2N^2$. Ti uvjeti osiguravaju da je $\frac{Q}{r} > N$.
2. Registri se inicijaliziraju u stanju

$$|\psi_0\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle |0\rangle.$$

3. Potrebno je konstruirati proroka za funkciju $f(x) = a^x \bmod N$ te ga primijeniti na stanje sustava. Sustav je u stanju

$$|\psi_1\rangle = \frac{1}{\sqrt{Q}} \sum_{x=0}^{Q-1} |x\rangle |f(x)\rangle.$$

4. Potrebno je primijeniti kvantnu Fourierovu transformaciju na prvi registar. Sustav je u stanju

$$|\psi_2\rangle = \frac{1}{Q}\sum_x \sum_y \omega^{xy}|y\rangle|f(x)\rangle.$$

gdje je $\omega = e^{\frac{2\pi i}{Q}}$.

5. Kada napravimo mjerenja registara, dobivamo vrijednost *y* iz prvog registra, te $f(x)$ iz drugog registra.
6. Pomoću ekspanzije kontinuiranih razlomaka na razlomku $\frac{y}{Q}$ radimo aproksimaciju te dobivamo $\frac{c}{r'}$, gdje je r' kandidat za *r*. Uvjeti za r' su:

$$A: r' < N$$
$$B: \left|\frac{y}{Q} - \frac{c}{r'}\right| < \frac{1}{2Q}$$

7. Ako je $f(x) = f(x + r') \Leftrightarrow a^{r'} \equiv 1 \pmod N$, r' je perioda funkcije *f(x)*. U protivnom, ponovi korak 6 koristeći za r' nove vrijednosti blizu *y*, ili vrijednosti koje su višekratnici korištenog broja r'.
8. U protivnom, idi na korak 1 i ponovi postupak ispočetka.

U konačnici svi koraci Shorovog algoritma su:

1. Izaberi broj $a < N$.
2. Izračunaj najvećeg zajedničkog djelitelja od *a* i *N*. Ukoliko je $\gcd(a, N) \neq 1$, vrati taj broj.
3. Izračunaj periodu *r* funkcije $f(x) = a^x \bmod N$.
4. Ako je *r* neparan, idi na korak 1.
5. Ako je $a^{\frac{r}{2}} \equiv -1(mod\ N)$, idi na korak 1.
6. Ako je $\gcd(a^{\frac{r}{2}} \pm 1, N)$ netrivijalni faktor od N. Algoritam terminira.

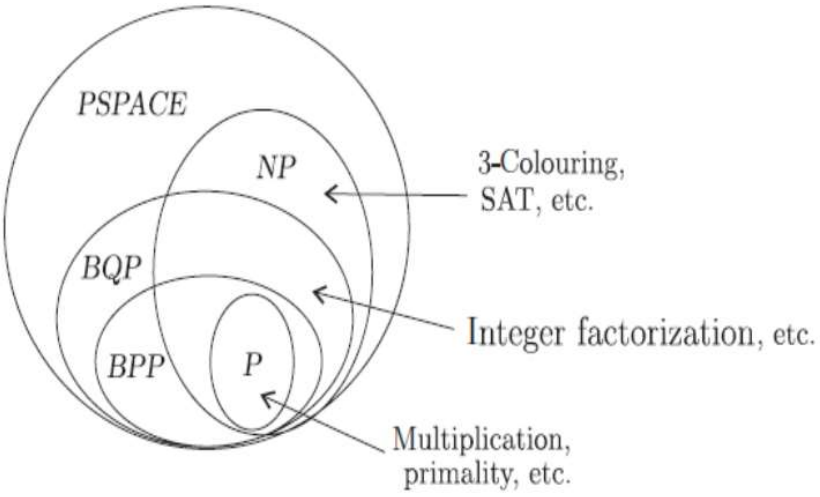
Vremenska složenost Shorovog algoritma ovisi o Fourierovoj kvantnoj transformaciji koja zahtijeva $O((\log N)^3)$ kvantnih operacija. Shorov algoritam je jedan od najjačih dokaza da kvantno računalo ima značajno više računalne snage od klasičnoga računala zato što rješava problem faktorizacije cijelih brojeva u polinomnom vremenu, dok klasičnom računalu treba eksponencijalno vrijeme za rješavanje istog problema.

5.11. Kvantna teorija složenosti

Kvantna teorija složenosti je dio računalne teorije složenosti koja se bavi proučavanjem klasa složenosti kod kvantnih računala [42][110].

Neke od najproučavanijih klasa složenosti su [78]:

- *P* - skup problema odlučivanja koji mogu biti riješeni u polinomnom vremenu na determinističkom Turingovom stroju.
- *NP* - skup problema odlučivanja koji mogu biti riješeni u polinomnom vremenu na nedeterminističkom Turingovom stroju.
- *BPP* - skup problema odlučivanja koji mogu biti riješeni u polinomnom vremenu na probabilističkom Turingovom stroju s najvećom vjerojatnošću greške od $\frac{1}{3}$.
- *BQP* - skup problema odlučivanja koji mogu biti riješeni u polinomnom vremenu na kvantnom računalu s najvećom vjerojatnošću greške od $\frac{1}{3}$.
- *PSPACE* - skup problema odlučivanja koji mogu biti riješeni na determinističkom Turingovom stroju upotrebom polinomne količine memorije.



Slika 22: Odnos klasa složenosti

Prethodna slika prikazuje odnos između spomenutih klasa složenosti. Treba napomenuti da spomenuti odnosi između klasa složenosti nisu dokazani, međutim vjeruje se da su takvi [2]. Možemo uočiti da klasa *BQP* uključuje širi skup problema nego klasa *BPP*, što proizlazi iz toga da interferencija između amplituda vjerojatnosti kod kvantnoga računala značajno utječe na brzinu odlučivanja zbog toga što se te amplitude vjerojatnosti mogu poništavati, dok to nije slučaj kod probabilističkog Turingovog stroja [97].

Važan dio kvantne teorije složenosti je složenost kvantnih pitanja (engl. *quantum query complexity*), odnosno broj postavljenih pitanja proroku koji je potreban algoritmu da riješi problem. Npr. Groverov algoritam ima složenost kvantnih pitanja $O(\sqrt{N})$.

Autor Yuri Ozhigov je u svom radu „*Quantum Computers Speed Up Classical with Probability Zero*“ [85] pokazao da postoje slučajevi kada kvantno računalo ne može ubrzati klasični algoritam koji se izvodi na samom kvantnom računalu, međutim računska složenost na kvantnom računalu za proizvoljni klasični algoritam u općem slučaju iznosi $\Omega(\sqrt{T})$, gdje je T broj koraka/operacija na klasičnom računalu.

5.12. Fizička realizacija kvantnoga računala

Predviđa se da bi kvantna računala trebala imati barem $10^2 \sim 10^3$ kvantnih bitova kako bi imalo smisla na njima rješavati one probleme, za koje postoje kvantni algoritmi koji

su brži od klasičnih algoritama. Izrada kvantnoga računala tih razmjera predstavlja izazov današnjoj tehnologiji i znanosti. DiVincenzo je predložio nekoliko neophodnih uvjeta koje fizički sustav mora zadovoljavati da bi bio kandidat za izvedbu kvantnoga računala [25].

1. Skalabilan fizički sustav s dobro opisanim kvantnim bitovima

Prvi uvjet je da sustav omogućuje pohranu informacija u kvantni registar, odnosno u skup kvantnih bitova. Najjednostavnija realizacija kvantnoga bita je dvorazinski kvantni sustav poput elektrona, spina jezgre ili dva međusobno okomita smjera polarizacije fotona. Sustav mora biti skalabilan i mora omogućiti dovoljan broj kvantnih bitova.

2. Sustav mora omogućiti inicijalizaciju stanja kvantnih bitova u neko jednostavno stanje poput $|00 \ldots 0\rangle$

Preduvjet gotovo svih kvantnih algoritama je neko jednostavno početno stanje, stoga fizički sustav mora omogućiti takvo stanje. Isto tako, fizički sustav se mora moći ponovno inicijalizirati u to stanje kako bi se mogao ponovo koristiti u obradi informacija.

3. Dugo vrijeme dekoherencije, puno duže nego vrijeme potrebno za operaciju kvantnih vrata

Dekoherencija je možda i ponajveći problem u izgradnji kvantnoga računala većih razmjera. Vrlo je teško izolirati kvantni sustav od svoje okoline koja narušava kvantnu informaciju unutar njega. Dekoherencija sama po sebi i nije toliko bitna, koliko je bitan omjer između vremena dekoherencije i vremena primjene operacije kvantnih vrata na kvantne bitove fizičkog sustava. Naime, što je vrijeme dekoherencije duže, to je više operacija moguće izvesti na fizičkom sustavu.

4. Univerzalni skup kvantnih vrata

Fizički sustav mora podržavati univerzalni skup kvantnih vrata kako bi svi kvantni algoritmi mogli biti realizirani.

5. Pojedinačni pristup kvantnim bitovima

Sustav mora omogućiti promjene stanja kao i mjerenje pojedinačnih kvantnih bitova.

6. Kvantni bitovi za komuniciranje

Sustav mora, osim kvantnih bitova za pohranu informacija, omogućavati i kvantne bitove za razmjenu informacija.

7. Mogućnost prijenosa kvantnih informacija između udaljenih lokacija

Sustav mora imati mogućnost slanja kvantnih informacija distribuiranim sustavima.

Autori Nakahara i Ohmi spominju 8 tehnologija za realizaciju kvantnoga računala u svojoj knjizi „*Quantum Computing: From Linear Algebra to Physical Realization*“ [81].

1. Nuklearna magnetska rezonanca
2. Ionske zamke
3. Zamka neutralnih atoma
4. Šupljine u kvantnoj elektrodinamici
5. Linearna optika
6. Kvantne točke
7. Josephsonovo čvorište
8. Elektroni na površini tekućeg helija

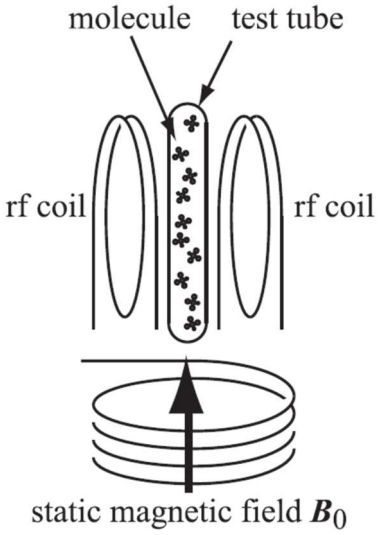
U nastavku će biti navedene neke od spomenutih tehnologija za fizičku realizaciju kvantnoga računala koje se često spominju u literaturi [13][23][81][82][83].

5.12.1. Nuklearna magnetska rezonanca

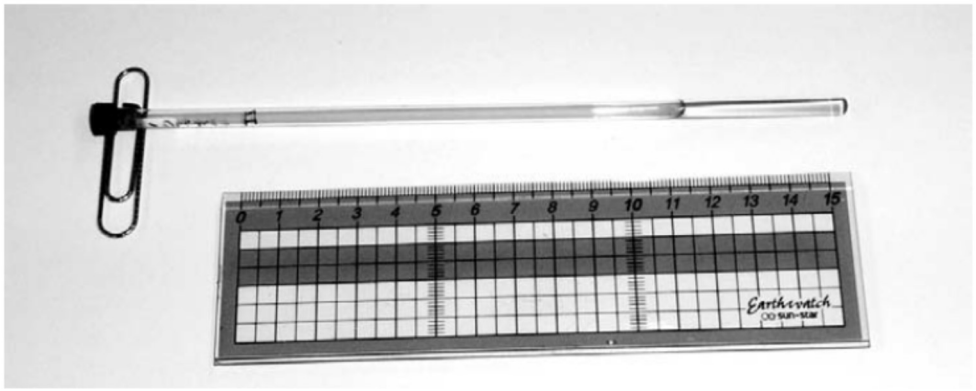
Nuklearna magnetska rezonanca (NMR) je jedna od najčešće upotrebljavanih tehnologija za izradu kvantnih računala. NMR upotrebljava spin u jezgri atoma molekule za

realizaciju kvantnih bitova. Za manipulaciju stanja kvantnih bitova, NMR koristi elektromagnetski puls, međutim problem kod te tehnologije je da ne može pristupiti individualnoj jezgri, već većem broju odjednom. Kvantni registar se sastoji od molekula s određenim brojem jezgri koje moraju biti u termičkoj ravnoteži, te se nalaze u staklenom cilindru na sobnoj temperaturi, pri čemu su svi spinovi jezgri u istom stanju. Međutim, upotrebom neunitarnih operacija spinovi se mogu pripremiti u neko od jednostavnijih stanja poput $|00 \ldots 0\rangle$ [81].

Stakleni cilindar se stavlja u spektrometar koji kontrolira spinove jezgri pomoću amplitude, frekvencije, faze i oblika elektromagnetskog vala. Sekvencu tih valova regulira algoritam koji se izvršava.



Slika 23: NMR tehnika [81]



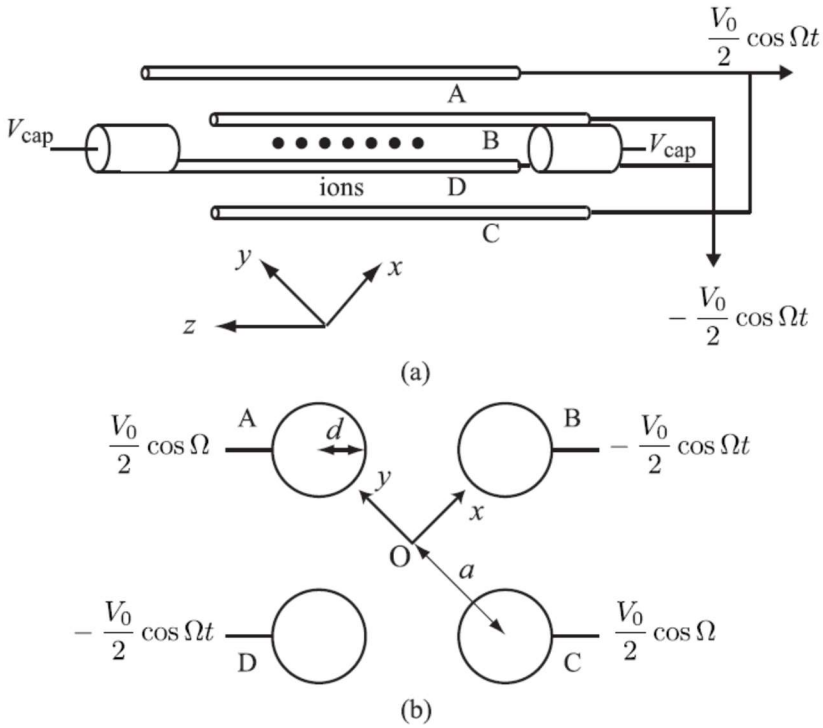
Slika 24: Molekule metanola u staklenom cilindru [81]

Mjerenja kvantnih bitova se radi pomoću FID (engl. *free induction decay*) tehnike, koja registrira signale koji su generirani kao posljedica neravnotežnog stanja u spinovima jezgri [72]. Vrijeme dekoherencije ovisi o molekulama koje se upotrebljavaju, a varira od $10^2 \sim 10^3\ s$. Trajanje operacije nad jednim kvantnim bitom traje otprilike $10^{-5}\ s$, dok trajanje operacije nad dva kvantna bita traje otprilike $10^{-2} \sim 10^{-1}\ s$. Nažalost, nedostatak ove tehnologije je što ona omogućuje realizaciju kvantnoga računala manjih razmjera, dok na većem broju kvantnih bitova nije primjenjiva zbog prevelikog utjecaja dekoherencije.

5.12.2. Ionske zamke

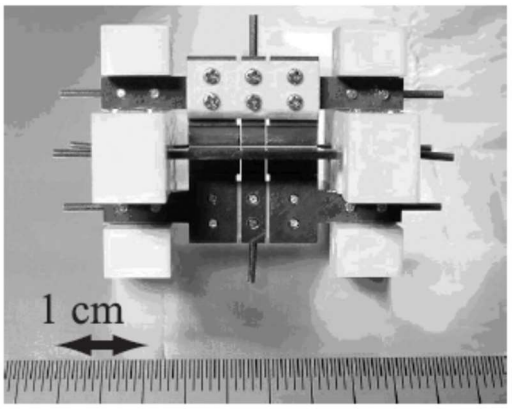
Tehnologija ionskih zamki je također jedna od najčešće spominjanih tehnologija za fizičku realizaciju kvantnih računala. Za kvantni bit se koriste energetske razine iona koji su „zarobljeni“ u ograničenom prostoru [99]. Ioni se adresiraju pomoću preciznih laserskih zraka, te se na taj način manipulira kvantnom informacijom smještenom unutar njih. Sama interakcija između nabijenih čestica regulirana je Coulumbovom silom. Mjerenja se obavljaju tehnikom „*electron shelving*“, pri čemu se stanje iona mjeri na način da se ion pogodi s fotonom, te ukoliko nakon toga ion emitira foton iste frekvencije kao i onaj kojim je pogođen, sustav je u jednom stanju, a u suprotnom je sustav u drugom stanju [30][113].

Inicijalizacija u neko od jednostavnijih stanja poput $|11 \ldots 1\rangle$ se radi pomoću preciznih laserskih zraka koje stavljaju ione u neko od pobuđenih stanja. Vrijeme dekoherencije kod kvantnih bitova u ovoj tehnologiji iznosi $10^{-1}\ s$. Jedan od načina realizacije ionske zamke je Paulova zamka, čija je logika prikazana na sljedećoj slici [81].



Slika 25: Paulova zamka

Ideja je da se ioni zarobe između četiri vodiča, čiji se naboj u vremenu dovoljno brzo izmjenjuje tako da ion ostaje u središtu. Fizička izvedba Paulove zamke se vidi na sljedećoj slici.

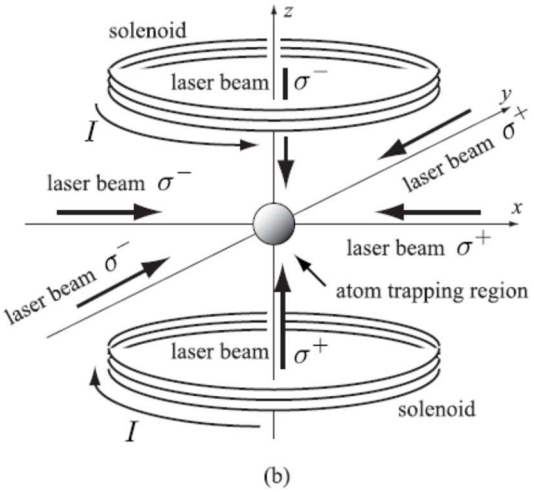


Slika 26: Fizička izvedba Paulove zamke

Očekuje se da je pomoću ove tehnologije moguće izgraditi kvantno računalo od otprilike 100 kvantnih bitova. Grupiranje više od stotinjak kvantnih bitova destabilizira sustav.

5.12.3. Zamka neutralnih atoma

Ideja ove tehnologije je da se neutralni atomi zarobe u određenom prostoru upotrebom laserskih zraka.



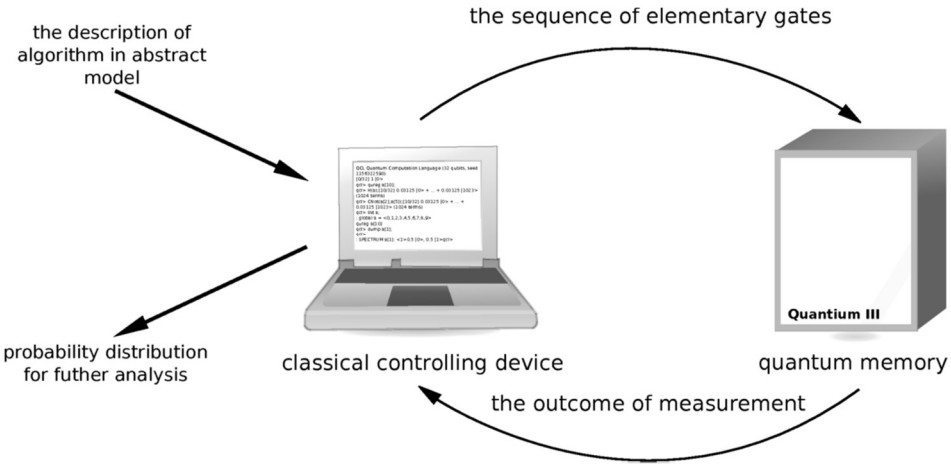
Slika 27: Zamka neutralnih atoma

Kada se neutralni atom giba određenim smjerom, on biva pogođen zrakom određenog lasera koji ga vraća u centar. Prednost upotrebe ove tehnologije je da okolina ima vrlo slab utjecaj na neutralne atome, te je vrijeme dekoherencije kod njih vrlo dugačko. Inicijalizacija početnog stanja se može napraviti na isti način kao i kod ionskih zamki, dakle pomoću preciznih laserskih zraka. Također, mjerenje se obavlja na isti način kao i kod tehnologije ionskih zamki — „*electron shelving*“ tehnikom. Predviđa se da bi se ovom tehnologijom moglo realizirati kvantno računalo veličine 10^6 kvantnih bitova [40].

5.13. Hibridni model klasičnoga i kvantnoga računala

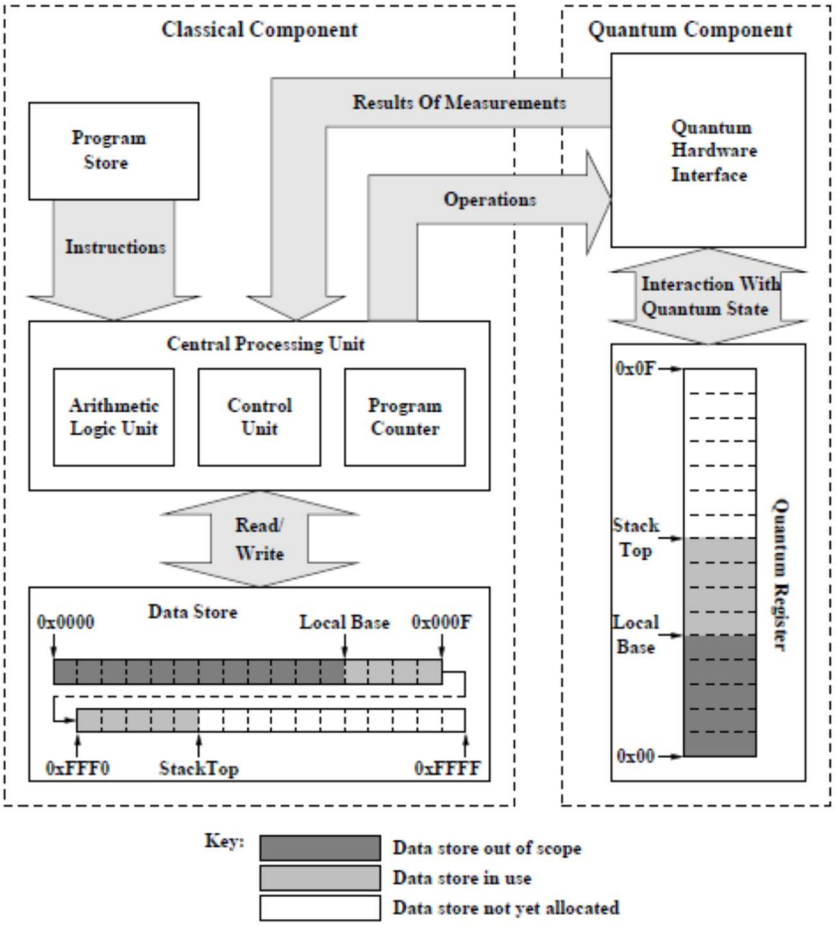
Predviđa se da će prvobitno najraširenji model korištenja kvantnoga računala biti klasični RAM (engl. *random access machine*) model s dodatnim resursom - kvantnom memorijom. Takav model se naziva QRAM (engl. *Quantum random access machine*) [36]. Dakako, konkretni model korištenja kvantnoga računala ovisi o konkretnom problemu i praktičnoj svrsi. Danas se kvantni algoritmi najčešće opisuju upotrebom matematičkih modela, te kombinacijom kvantnoga i klasičnoga sklopovlja. QRAM model pomoću

klasičnoga računala kontrolira stanje unutar kvantne memorije. Shema takvog modela je prikazana na sljedećoj slici.



Slika 28: QRAM model [36][41]

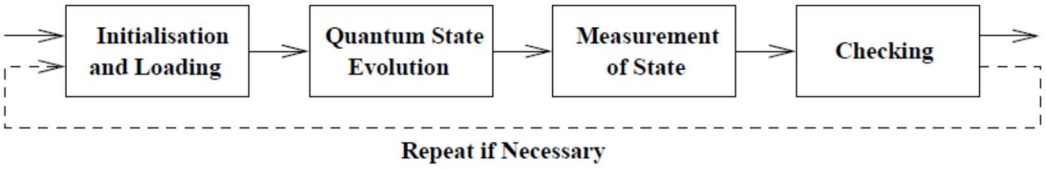
Ideja je da se na klasičnom računalu izvršava algoritam koji primjenjuje unitarne operacije na kvantnu memoriju, odnosno stanje kvantnoga sustava. Povratno, klasično računalo iz kvantne memorije prihvaća rezultate mjerenja kvantnoga stanja. Prema prijedlogu autora Rajagopal Nagarajan i Nikolaos Papanikolaou, skica SQRAM (engl. Sequence Quantum Random Access Machine) računala izgleda [80]:



Slika 29: SQRAM model

Klasična komponenta se sastoji od dvije zasebne memorijske lokacije, jedna je za program, a druga za podatke, što je ipak malo drugačije od von Neumannovog modela, gdje se podaci i program nalaze unutar iste memorijske lokacije. *CPU* jedinica preko programskog brojača indeksira instrukciju koja se izvodi. *CPU* također sadrži aritmetičko logičku jedinicu za evaluaciju klasičnih izraza, te kontrolnu jedinicu za kontroliranje drugih operacija. Također, klasična komponenta koristi stog za alokaciju varijabli i evaluaciju izraza. Operativni ciklus unutar *SQRAM*-a izgleda tako da su na početku vrijednosti programskog brojača i stoga inicijalizirane na 0. Kako se program izvodi, programski brojač se povećava i izvodi instrukcije koje su zapisane u memoriji za instrukcije. Program završava kada programski brojač prijeđe određenu vrijednost koja reprezentira kraj instrukcija.

Kvantna komponenta *SQRAM*-a se sastoji od kvantnoga registra i kvantnoga hardverskog sučelja (*QHI*), koje prima instrukcije od *CPU*-a i manipulira stanjima unutar kvantnih bitova. Sljedeća slika prikazuje tipične faze izvršavanja kvantnoga algoritma.



Slika 30: Faze izvršavanja kvantnoga algoritma

U prvoj fazi, *QHI* inicijalizira kvantne bitove u stanju $|00 \dots 0\rangle$, te primjenjuje transformacije kako bi doveo trenutno stanje kvantnih bitova u željeno početno stanje. U drugoj fazi *QHI* primjenjuje korake kvantnoga algoritma, dok u trećoj fazi *QHI* obavlja mjerenje kvantnih bitova. Rezultat mjerenja svakog kvantnoga bita je 0 ili 1, što se u konačnici vraća *CPU*-u. U zadnjoj fazi neposredno prije vraćanja rezultata *CPU*-u, *QHI* provjerava valjanost rezultata te ponavlja postupak ukoliko je to potrebno, s obzirom da su zbog kvantne dekoherencije ili probabilističke prirode kvantnih algoritama moguće pogreške.

Već i danas postoje praktične primjene hibridnog modela klasičnoga i kvantnoga računala. Primjeri takve primjene su algoritmi za optimizaciju vodovodne mreže te optimizaciju radioterapije u medicini [52].

5.14. Simulacija kvantnoga računala na klasičnom računalu

Klasično računalo može simulirati kvantno računalo ukoliko mu se dodijeli dovoljno resursa, međutim ne efikasno. Ukoliko bi klasično računalo moglo efikasno simulirati kvantno računalo, računalna snaga klasičnoga i kvantnoga računala bi bila podjednaka. Naime, za spremanje kvantnoga stanja potrebno je puno više memorije nego što je potrebno za spremanje klasičnoga stanja. Također za obradu kvantnoga stanja je potrebno puno više računalnih resursa nego što to zahtijeva klasično stanje.

Stanje kvantnomehaničkog sustava od N kvantnih bitova se opisuje vektorom u kompleksnom vektorskom prostoru dimenzija 2^N, što znači da je u klasično računalo potrebno pohraniti 2^N kompleksnih vrijednosti ukoliko želimo sačuvati stanje kvantnomehaničkog sustava od N kvantnih bitova. Također treba napomenuti da u praksi kod spremanja kompleksnih vrijednosti u klasično računalo uvijek dolazi do pogreške u zaokruživanju zbog ograničenosti memorijskih resursa.

Svaki korak računanja primjene kvantnih vrata na kvantno stanje zahtijeva primjenu kompleksne matrice dimenzija $2^N \times 2^N$ na vektor stanja koji ima 2^N elemenata iz kompleksne domene, dakle to je u najgorem slučaju 4^N množenja kompleksnih brojeva, te 4^{N-1} zbrajanja kompleksnih brojeva, što je dosta zahtjevan posao za klasično računalo. N je, dakako, broj kvantnih bitova.

Danas postoji popriličan broj simulatora kvantnoga računala na klasičnom računalu. Prema izvoru http://www.quantiki.org/wiki/List_of_QC_simulators [58] postoji oko stotinjak simulatora kvantnih računala koji su dostupni preko Interneta. S obzirom da kvantno računalo još nije komercijalni proizvod, dostupni simulatori kvantnoga računala imaju veliki značaj u razvijanju, testiranju i učenju algoritama i logike kvantnih računala. Složenost simulacija koje simulatori kvantnoga računala izvode značajno variraju ovisno o samom simulatoru. Neki simulatori simuliraju samo diskretnu logiku rada kvantnoga računala, dok drugi simuliraju i fizičku realizaciju kvantnog sklopovlja [11][20]. Simulatore kvantnih računala možemo klasificirati u pet kategorija [20][78][98]:

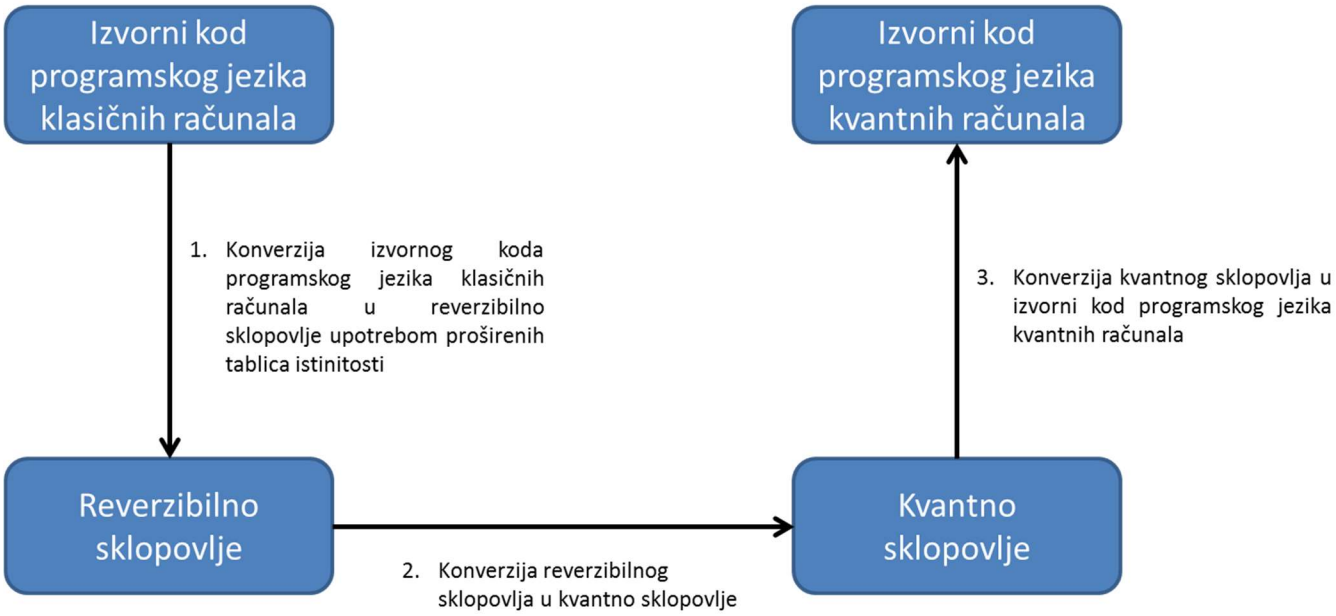
1. Programski jezici za kvantna računala - izražavaju semantiku računanja na apstraktan način te automatski generiraju elementarne operacije za upravljanje kvantnim računalom. Primjeri programskih jezika za kvantna računala su QCL [84], QPL [94], LanQ [79], Scaffold [67], QASM [67], QML [37].
2. Kvantni *compileri* - za ulaz primaju unitarnu transformaciju, a za izlaz daju niz elementarnih operacija nad jednim i dva kvantna bita koje izvršavaju ulaznu unitarnu transformaciju. Primjeri kvantnih compilera su Qubiter [100], Qcompiler [15], OptQC [73], ScaffCC [68].
3. Simulatori sklopovlja kvantnoga računala - simulirajući kvantno sklopovlje osiguravaju programsko okruženje za simulaciju kvantnih računala. Primjeri simulatora sklopovlja kvantnoga računala su QCAD [107], QC-Lib [14], Libquantum [55].
4. Simulatori fizičke realizacije kvantnoga računala - uzimaju u obzir dinamiku sustava u vremenu. Primjeri simulatora fizičke realizacije kvantnoga računala su QCE [58], QSS [90].
5. Simulatori kvantnoga računala za edukativne svrhe - simuliraju određeni algoritam kvantnoga računala. Primjeri simulatora kvantnoga računala za edukativne svrhe su Quantum Search Simulator [14], Grover's algorithm simulator [69], Shor's algorithm simulator [58].

6. Prevođenje izvornoga programskog koda klasičnoga računala u izvorni programski kod kvantnoga računala

Pored specifičnih programskih jezika za klasična računala te specifičnih programskih jezika za kvantna računala, danas postoje i hibridni programski jezici koji objedinjuju klasičnu i kvantnu logiku na jednom mjestu. U hibridnim jezicima moguće je razvijati programe u kojima se neki dijelovi koda izvršavaju na klasičnom računalu dok se drugi dijelovi koda na kvantnome računalu. Vrlo je izvjesno da će se hibridni jezici koristiti u budućnosti iz razloga što će i sama arhitektura budućih računala vrlo izvjesno biti hibridna [65]. Već i danas postoje računala koje se temelje na hibridnoj arhitekturi, npr. kompanija Rigetti Computing razvila je programsko rješenje pomoću kojeg je moguće razvijati i izvršavati hibridne programe na hibridnom računalu koje se nalazi u cloud okruženju [66]. Neki od hibridnih jezika koji se koriste u praksi su Quil [92] te Q# [61]. Iako hibridni jezici jesu budućnost razvoja programskih rješenja oni ipak ne rješavaju problem prevođenja izvornih kodova postojećih programa za klasična računala u izvorne kodove programa za kvantna računala što je tema ovog rada. U dostupnoj literaturi u vrijeme pisanja ovog rada ne postoji niti jedan rad drugih autora koji obrađuje temu prevođenja izvornoga koda klasičnih računala u izvorni kod kvantnih računala. Smatram da je tome razlog tehnologija za fizičku realizaciju kvantnih računala koja je još u ranoj fazi, te još nije spremna za realizaciju kvantnoga računala većih razmjera. Kako kvantno računalo još nije komercijalni proizvod, tako realno još nema potrebe za rješenjem koje će automatski pretvarati izvorni programski kod za klasična računala u izvorni programski kod za kvantna računala. Međutim, možda tehnologija kvantnih računala jednog dana zaista bude dovoljno zrela i kvantno računalo postane komercijalni proizvod. U tom slučaju vjerujem da će ova problematika postati aktualna. U ovom dijelu rada pobliže ću opisati istraživanje koje sam proveo, a vezano je uz problematiku prevođenja izvornoga programskog koda klasičnoga računala u izvorni programski kod kvantnoga računala. U sklopu tog istraživanja predstavit ću pristup koji sam koristio u realizaciji prevođenja te rezultate testiranja prevođenja koje se temeljilo na predloženom pristupu. Uz ovaj rad bit će priložen i izvorni programski kod prevoditelja koji je implementiran na opisanim načelima.

6.1. Pristup prevođenju izvornoga programskog koda klasičnoga računala u izvorni programski kod kvantnoga računala

Slika ispod prikazuje korake koji su potrebni da bi se izvorni programski kod klasičnoga računala preveo u izvorni programski kod kvantnoga računala. U nastavku će svaki od tih koraka biti i pobliže opisan [8].

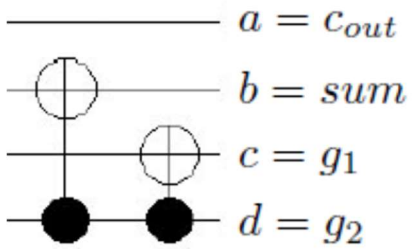


Slika 31: Pristup prevođenju izvornoga programskog koda klasičnoga računala u izvorni programski kod kvantnoga računala

6.1.1. Prevođenje izvornoga koda programskog jezika klasičnoga računala u reverzibilno sklopovlje upotrebom proširenih tablica istinitosti

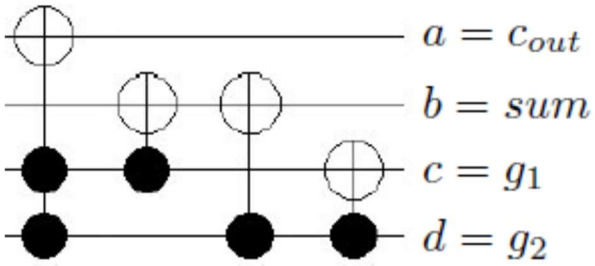
Svaki determinirani program koji se izvodi na klasičnom digitalnom računalu za ulaze i izlaze ima nizove bitova, te se može prikazati pomoću klasičnoga logičkog sklopovlja koje ne mora biti reverzibilno. Logika logičkog sklopovlja, pa tako i logika programa klasičnoga računala, može se prikazati preko tablice istinitosti. U poglavlju 5.2. smo vidjeli da se svaka tablica istinitosti može proširiti da bude reverzibilna dodavanjem bitova na ulaze i izlaze. Na temelju reverzibilne tablice istinitosti moguće je generirati reverzibilno sklopovlje na način kako su to opisali autori D. M. Miller, D. Maslov, and G. W. Dueck u svojem radu „*A transformation based algorithm for reversible logic synthesis*“ [77]. Ideja je da se reverzibilno

sklopovlje generira na način da se prolazi redak po redak u proširenoj tablici istinitosti, te da se u reverzibilno sklopovlje dodaju reverzibilni sklopovi koji ulaze za taj redak pretvaraju u izlaze tog retka. Reverzibilni sklopovi se dodaju od kraja prema početku reverzibilnog sklopovlja. Kada se dodaju novi sklopovi u sklopovlje, oni ne smiju mijenjati već izmijenjene retke. U nastavku je opisan postupak na primjeru proširene tablice za potpuno zbrajalo koja je prikazana u tablici 6 u poglavlju 5.2. U prvom retku proširene tablice istinitosti svi ulazi i izlazi su 0, te u tom slučaju nije potrebno ništa mijenjati pa možemo prijeći na drugi redak. U drugom retku postoje razlike na drugom i trećem bitu između ulaza i izlaza, stoga drugi i treći bit moraju biti negirani. S obzirom da je zadnji bit ujedno i kontrolni, ovo dodavanje ne utječe na logiku prethodnog retka. Stavljamo dvoja *CNOT* vrata tako da reverzibilno sklopovlje izgleda ovako:



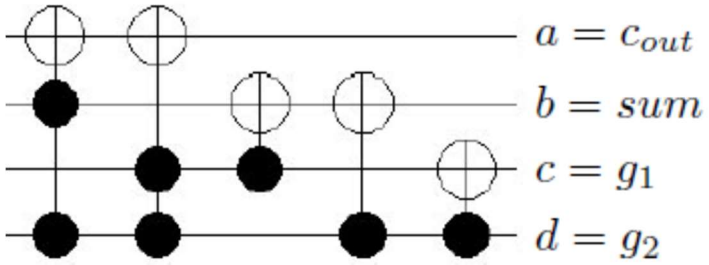
Slika 32: Reverzibilno sklopovlje nakon obrade prva dva retka proširive tablice istinitosti za potpuno zbrajalo

Za treći redak trebamo dodati nova *CNOT* vrata koja utječu na drugi bit u zavisnosti o trećem bitu. Za četvrti redak trebamo dodati *Toffolijeva* vrata koja su kontrolirana pomoću trećeg i četvrtog bita, a utječu na prvi bit. Reverzibilni sklop nakon obrađenog četvrtog retka izgleda ovako:



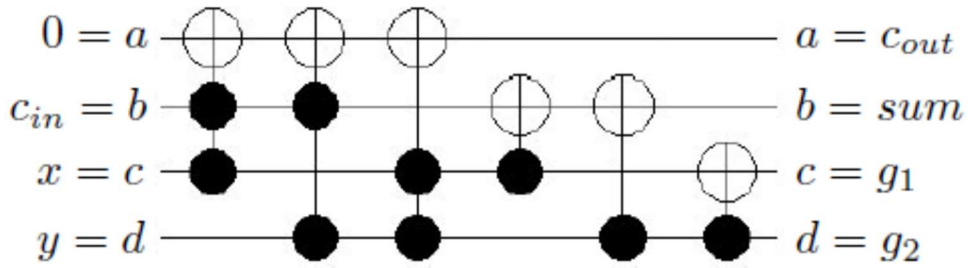
Slika 33: Reverzibilno sklopovlje nakon obrade prvih četiri retka proširive tablice istinitosti za potpuno zbrajalo

Za peti redak nije potrebno ništa mijenjati, dok je za šesti je potrebno dodati *Toffolijeva* vrata koja su kontrolirana od strane drugog i četvrtog bita, a utječu na prvi bit. Reverzibilni sklop nakon obrađenog šestog retka izgleda ovako:



Slika 34: Reverzibilno sklopovlje nakon obrade prvih šest redaka proširive tablice istinitosti za potpuno zbrajalo

Za sedmi redak potrebno je dodati jedna *Toffolijeva* vrata koja utječu na prvi bit, a kontrolirana su pomoću drugog i trećeg bita. Za osmi redak nije potrebno dodavati niti jedna vrata, pa tako reverzibilni sklop u konačnici izgleda:



Slika 35: Reverzibilno sklopovlje za potpuno zbrajalo

Potrebno je napomenuti da je reverzibilno sklopovlje moguće na sličan način generirati i manipulacijom ulaznih bitova što su i autori D. M. Miller, D. Maslov, and G. W. Dueck u svojem radu i opisali. Prilikom implementacije prevoditelja koristila se kombinacija ta dva pristupa što je ujedno i preporuka autora MMD algoritma.

6.1.2. Prevođenje reverzibilnog sklopovlja u kvantno sklopovlje

Kao što je spomenuto u poglavlju 5.2., *Toffolijeva* vrata su univerzalna vrata za reverzibilno sklopovlje, te se mogu realizirati u kvantnom sklopovlju, dakle bilo koje

reverzibilno sklopovlje moguće je pretvoriti u kvantno sklopovlje. Dakako, prilikom generiranja kvantnoga na temelju reverzibilnog sklopovlja, bitno je da generirano kvantno sklopovlje ima što manji trošak po nekom kriteriju. Neki od kriterija, odnosno troškova, spomenuti su u poglavlju 5.10. Jedan od načina generiranja kvantnog sklopovlja je da se reverzibilno sklopovlje izravno pretvori u kvantno sklopovlje ukoliko se reverzibilno sklopovlje sastoji od reverzibilnih vrata koja imaju svoju izravnu reprezentaciju u kvantnom sklopovlju (npr. *Toffolijeva* i *CNOT* vrata). Jedan od algoritama koji to osiguravaju je opisan u poglavlju 6.1.1. Isto tako, ukoliko se reverzibilno sklopovlje sastoji od reverzibilnih vrata koja imaju svoju izravnu reprezentaciju u kvantnom sklopovlju, tada je moguće za takvo reverzibilno sklopovlje izračunati unitarnu matricu, koju je pak moguće dekomponirati na faktore pomoću algoritma za dekompoziciju unitarnih matrica. Na temelju dobivenih faktora se u konačnici generira kvantno sklopovlje. Neki od algoritama za dekompoziciju unitarnih matrica su Daskininov i Kaisov algoritam [19], Lukacov i Perkowskiov algoritam [74], kosinus-sinus dekompozicija (CSD) [101], QR dekompozicija [34]. Kosinus-sinus dekompozicija (CSD) i QR dekompozicija su najčešće spominjani algoritmi u literaturi te će biti opisani u ovom radu.

6.1.2.1. CS dekompozicija unitarne matrice

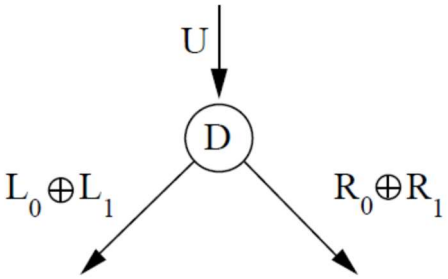
Pretpostavimo da je U unitarna matrica dimenzija $2^N \times 2^N$, gdje je N prirodni broj. Algoritam *CS* dekompozicije dekomponira matricu U u formu [101]

$$U = \begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix} D \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix},$$

gdje su matrice L_0, L_1, R_0, R_1 unitarne matrice dimenzija $2^{N-1} \times 2^{N-1}$, a D matrica je u formi

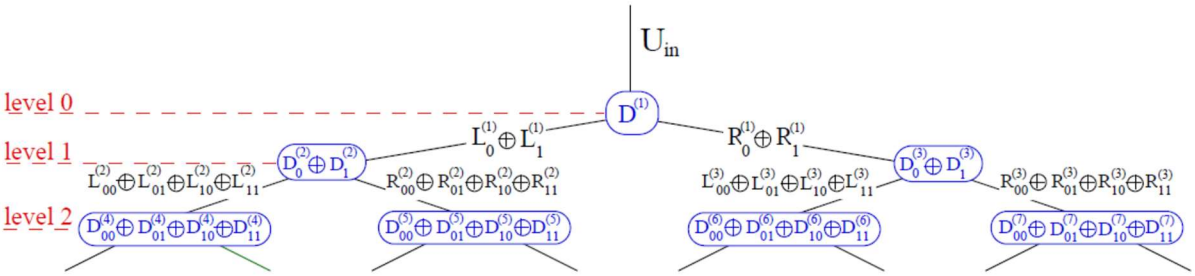
$$D = \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix},$$

gdje su $D_{01} = diag\left(S_1, S_2, \dots, S_{\frac{N}{2}}\right), D_{10} = -D_{01}, D_{00} = D_{11} = diag\left(C_1, C_2, \dots, C_{\frac{N}{2}}\right)$, za sve $i \in Z_{1,\frac{N}{2}}$, $C_i = \cos\theta_i$ i $S_i = \sin\theta_i$ za neki kut θ_i. Dakle, $U = (L_0 \oplus L_1)e^{i\sigma Y \otimes \theta}(R_0 \oplus R_1)$, gdje je $\theta = diag\left(\theta_1, \theta_2, \dots, \theta_{\frac{N}{2}}\right)$.



Slika 36: CS dekompozicija

Rekurzivna primjena CS dekompozicije čini binarno stablo prikazano na sljedećoj slici:



Slika 37: Binarno stablo dobiveno rekurzivnom primjenom CS dekompozicije

Dakle, algoritam počinje s dekompozicijom matrice U_{in} koja ima dimenzije $2^N \times 2^N$, gdje je N prirodni broj. Rezultat dekompozicije je matrica $D^{(1)}$, dvije unitarne matrice $L_0^{(1)}$ i $L_1^{(1)}$ dimenzija $2^{N-1} \times 2^{N-1}$ na lijevoj strani, te dvije unitarne matrice $R_0^{(1)}$ i $R_1^{(1)}$ dimenzija $2^{N-1} \times 2^{N-1}$ na desnoj strani. Tada se ponovno primjenjuje *CS* dekompozicija na spomenute četiri matrice, te se ukupno dobiva 8 L i 8 R unitarnih matrica dimenzija $2^{N-2} \times 2^{N-2}$ s pripadajućim D matricama. Rekurzivna primjena *CS* dekompozicije se izvršava sve dok L i R matrice ne budu dimenzija 2×2. Jedan od algoritama za realizaciju *CS* dekompozicije je

opisao Van Loan u svojem radu „*Computing the CS and the generalized singular value decompositions*“ [105].

Kada dekompozicija završi, potrebno je pretvoriti dobivene matrice u kvantne sklopove. Pretpostavimo da je matrica unitarna matrica U dimenzija 4×4, te da smo *CS* dekompozicijom U matrice dobili

$$U = \begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix} D \begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix}.$$

$L_0, L_1, R_{0,} R_1$ su unitarne matrice dimenzija 2×2, dok je matrica D dimenzija 4×4. Matrica $\begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix}$ se može pisati kao

$$\begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix} = \begin{bmatrix} L_0 & & 0 & 0 \\ & L_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & & L_1 \\ 0 & 0 & & \end{bmatrix},$$

odnosno matrica $\begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix}$ se može pisati kao

$$\begin{bmatrix} R_0 & 0 \\ 0 & R_1 \end{bmatrix} = \begin{bmatrix} R_0 & & 0 & 0 \\ & R_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & & R_1 \\ 0 & 0 & & \end{bmatrix}.$$

Možemo zaključiti da je matrice L_0, L_1, R_0, R_1 moguće realizirati kao kontrolna vrata kako je to opisano u poglavlju 5.9.1.

Matrica D se sastoji od 4 dijagonalne matrice $D_{00}, D_{01}, D_{10}, D_{11}$.

$$D = \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix},$$

što još možemo pisati kao

$$D = \begin{bmatrix} D_{000} & 0 & D_{010} & 0 \\ 0 & D_{001} & 0 & D_{011} \\ D_{100} & 0 & D_{110} & 0 \\ 0 & D_{101} & 0 & D_{111} \end{bmatrix},$$

odnosno

$$D = \begin{bmatrix} D_{000} & 0 & D_{010} & 0 \\ 0 & 1 & 0 & 0 \\ D_{100} & 0 & D_{110} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & D_{001} & 0 & D_{011} \\ 0 & 0 & 1 & 0 \\ 0 & D_{101} & 0 & D_{111} \end{bmatrix}.$$

Vidimo da je matrica D zapravo produkt dvije dvorazinske matrice koje je moguće realizirati na način kako je to opisano u poglavlju 5.9.2.2.

6.1.2.2. QR dekompozicija unitarne matrice

QR dekompozicija dekomponira matricu A na produkte QR, gdje je Q ortogonalna matrica, a R gornja trokutasta matrica [34]. Za realizaciju QR dekompozicije mogu se upotrijebiti Givensove rotacije. Givensova rotacija se reprezentira preko matrice

$$G(i,j,\theta) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & \bar{c} & & \bar{s} & 0 \\ \vdots & & \ddots & \vdots & \\ 0 & -s & \cdots & c & 0 \\ 0 & 0 & & 0 & 1 \end{bmatrix}.$$

Slika 38: Givensova matrica

Dakle, Givensova matrica ima elemente različite od nule ako vrijedi

$$g_{kk} = 1 \quad za\ k \neq i,j$$

$$g_{ii} = \bar{c}\ za\ i < j$$

$$g_{ii} = c\ za\ i > j$$

$$g_{jj} = c\ za\ i < j$$

$$g_{jj} = \bar{c}\ za\ i > j$$

$$g_{ji} = -s\ za\ i > j$$

$$g_{ij} = s \quad za\ i < j.$$

Produkt $G(i,j,\theta)x$ predstavlja rotaciju vektora x u smjeru obrnutom od kazaljke na satu za kut od θ radijana u podprostoru koji je razapet i,j vektorima. Ukoliko Givensova matrica množi neku matricu A s lijeve strane, ona utječe na i-ti i j-ti redak u matrici A. Sagledajmo idući problem, gdje je zadan vektor čiji elementi su a i b, te je potrebno pronaći takve c i s da vrijedi

$$\begin{bmatrix} \bar{c} & \bar{s} \\ -s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

To možemo riješiti na sljedeći način:

$$r \leftarrow \sqrt{\|a\|^2 + \|b\|^2}$$

$$c \leftarrow \frac{a}{r}$$

$$s \leftarrow \frac{b}{r}.$$

Givensove matrice možemo upotrijebiti za izračunavanje QR dekompozicije na način da Givensove matrice nuliraju elemente iz donjeg trokuta matrice A. Uzastopnom primjenom Givensove matrice dobit ćemo matricu Q. Pretpostavimo da imamo unitarnu matricu U dimenzija 4×4

$$U = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ a & * & * & * \\ b & * & * & * \end{bmatrix},$$

te da želimo nulirati element b u prvom stupcu, četvrtom retku. Tada ćemo napraviti rotaciju

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \bar{c} & \bar{s} \\ 0 & 0 & -s & c \end{bmatrix},$$

gdje su

$$r \leftarrow \sqrt{\|a\|^2 + \|b\|^2}$$

$$c \leftarrow \frac{a}{r}$$

$$s \leftarrow \frac{b}{r}.$$

Kada s desne strane pomnožimo matricu U s matricom G_1 dobivamo

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \bar{c} & \bar{s} \\ 0 & 0 & -s & c \end{bmatrix} \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ a & * & * & * \\ b & * & * & * \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ r & *' & *' & *' \\ 0 & *' & *' & *' \end{bmatrix}.$$

Uzastopnom primjenom Givensovih matrica $G_1 \dots G_n$ dobit ćemo $G_n \dots G_1 U = R$, gdje je U prvobitna matrica. Iz toga proizlazi da je $U = G_1^{-1} \dots G_n^{-1} R$. S obzirom da faktori unitarne matrice također moraju biti unitarni, tako i R mora biti unitarna. Jedina matrica koja je gornje trokutasta i unitarna je dijagonalna matrica. Također, vrijednosti na dijagonali matrice R moraju biti pozitivne, s obzirom da se radi o dekompoziciji unitarne matrice U. Stoga zaključujemo da matrica R mora biti jedinična matrica. Dakle, konačna dekompozicija je

$$U = G_1^{-1} \dots G_n^{-1}$$

Maksimalni broj Givensovih matrica prilikom ovakvog izračuna QR dekompozicije je $\frac{N(N-1)}{2}$. S obzirom da su Givensove matrice dvorazinske matrice, moguće ih je pretvoriti u kvantne sklopove na način kako je to opisano u poglavlju 5.8.2.2.

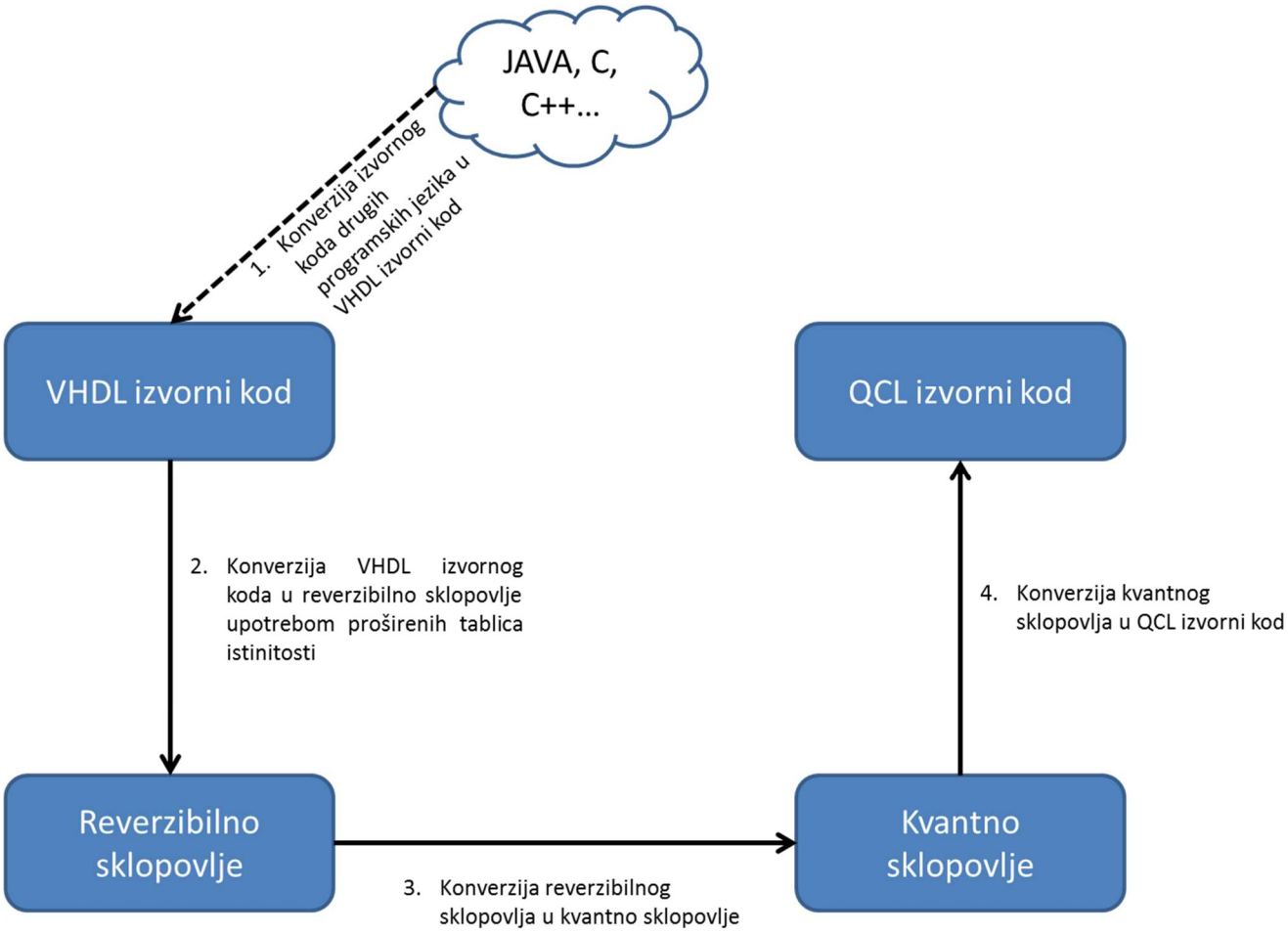
6.1.3. Prevođenje kvantnog sklopovlja u izvorni kod programskog jezika kvantnoga računala

Zadnji korak u predloženom pristupu je prevođenje generiranog kvantnog sklopovlja u izvorni kod programskog jezika kvantnoga računala. To ne bi trebalo predstavljati veliki problem zato što je većina naredbi programskih jezika za kvantna računala na sklopovskoj

razini, pa se ovaj korak svodi na pridruživanje kvantnih vrata iz kvantnog sklopovlja na konkretnu naredbu iz programskog jezika za kvantna računala. U konačnici, nakon dobivenog izvornog koda za programski jezik kvantnoga računala, isti je moguće izvršiti na kvantnome računalu i dobiti izlazni rezultat za zadani ulaz.

6.2. Implementacija prevođenja izvornoga programskog koda klasičnoga računala u izvorni programski kod kvantnoga računala

U ovom dijelu rada opisat ću kako sam realizirao konkretnu implementaciju prevoditelja na temelju opisanog pristupa u prethodnom poglavlju. Dakle, cilj je bio implementirati prevoditelja koji će prevoditi izvorni kod programskog jezika klasičnoga računala - VHDL u izvorni kod programskog jezika za kvantna računala - QCL. Implementacija je napravljena u programskom jeziku C++ (izvorni kod je na CD-u koji je priložen uz ovaj rad).

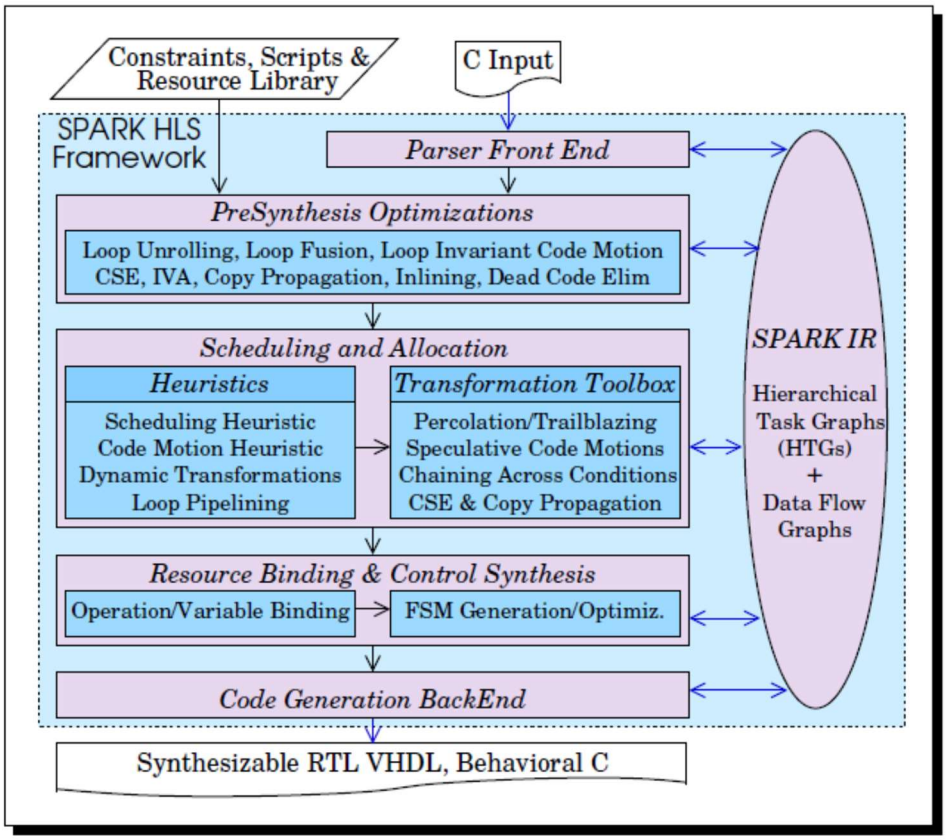


Slika 39: Koraci realizirane implementacije

6.2.1. Prevođenje izvornoga koda drugih programskih jezika klasičnoga računala u VHDL izvorni kod

VHSIC Hardware Description Language (VHDL) je programski jezik klasičnih računala za opisivanje digitalnih sustava [3]. Za mnoge druge programske jezike klasičnih računala (npr. Java, C) postoje razvijeni prevoditelji, koji njihov izvorni kod pretvaraju u izvorni kod VHDL-a. Jedan od takvih prevoditelja je i SPARK [48] koji izvorni kod programskog jezika C pretvara u izvorni programski kod VHDL-a. Nakon realizacije VHDL → QCL prevoditelja možemo iskoristiti već postojeće prevoditelje izvornih kodova drugih programskih jezika klasičnoga računala u VHDL kod s konačnim ciljem prevođenja izvornoga koda drugih programskih jezika klasičnoga računala u izvorni kod QCL-a.

Prevođenje C koda u VHDL kod upotrebom SPARK prevoditelja se odvija u četiri koraka koji su prikazani na sljedećoj slici [39].



Slika 40. Koraci SPARK prevoditelja

1. **Optimiziranje izvornog koda prije sinteze**

   U ovom koraku prevoditelj radi transformaciju izvornog C koda prilikom koje eliminira one dijelove koda koji nemaju utjecaj na rezultat, eliminira petlje te eliminira nepotrebne i redundantne operacije.

2. **Planiranje i alokacija**

   Prevoditelj izvršava napredniju transformaciju u kojoj planira redoslijed izvršavanja operacija. Pri tome pokušava postići paralelizam na dijelovima koda gdje je to moguće pri čemu uzima u obzir količinu resursa koja mu je dodijeljena od strane dizajnera.

3. **Povezivanje sa resursima i kontrola sinteze**

U ovom koraku prevoditelj vrši mapiranje operacija sa konkretnim funkcionalnim jedinicama te mapiranje varijabli sa registrima. Također, u ovom koraku se definira i kontrolna jedinica koja će kontrolirati takt izvođenja operacija.

4. **Generiranje koda**

   U posljednjem koraku prevoditelj generira VHDL kod na temelju sintetiziranog dizajna u prethodnom koraku.

Npr., uzmimo sljedeći C kod koji provjerava da li je zadani broj ujedno i savršeni broj

```
#include <stdio.h>
int main(int inNumber)
{
   int reminder, sumDivider = 0, i = 1;

   while(i<=(inNumber/2)){
      reminder = inNumber % i;
           if (reminder == 0)
      {
         sumDivider = sumDivider + i;
      }
           i++;
   }

   if (sumDivider == inNumber)
      return 1;
   else
      return 0;

}
```

Pomoću SPARK prevoditelja prevodimo C kod u VHDL kod te dobivamo

```
-- Automatically generated by the SPARK Parallelizing High-Level Synthesis Framework
-- Mon Mar 19 19:52:00 2018, source file : perfect.c

-- 'SPARK' should be defined as the user package
PACKAGE spark_pkg is
  TYPE integer_vector IS ARRAY ( NATURAL RANGE <>) OF integer;
  TYPE boolean_vector IS ARRAY ( NATURAL RANGE <>) OF boolean;
  FUNCTION integer_wired_or ( arr_int : integer_vector ) RETURN integer;
  FUNCTION boolean_wired_or ( arr_bool : boolean_vector ) RETURN boolean;
  SUBTYPE wiredOrInt IS integer_wired_or integer;
  SUBTYPE wiredOrBoolean IS boolean_wired_or boolean;
END spark_pkg;


library IEEE;
use IEEE.std_logic_1164.all;
```

```
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;

PACKAGE BODY spark_pkg IS
 FUNCTION integer_wired_or ( arr_int : integer_vector ) RETURN integer is
  -- pragma resolution_method wired_or
  variable i : integer;
  variable returnVal : std_logic_vector(0 downto 0);
  variable arr_int_std_logic_vec : std_logic_vector(0 downto 0);
 BEGIN
  returnVal := (others => '0');
  for i in arr_int'range loop
   arr_int_std_logic_vec := conv_std_logic_vector(arr_int(i), 1);
   returnVal := returnVal or arr_int_std_logic_vec;
  end loop;
  RETURN conv_integer(returnVal);
 END integer_wired_or;

 FUNCTION boolean_wired_or ( arr_bool : boolean_vector ) RETURN boolean is
  -- pragma resolution_method wired_or
  variable i : integer;
  variable returnVal : boolean;
 BEGIN
  returnVal := FALSE;
  for i in arr_bool'range loop
   returnVal := returnVal or arr_bool(i);
  end loop;
  RETURN returnVal;
 END boolean_wired_or;
end spark_pkg;

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;

library work;
use work.spark_pkg.all;


ENTITY main IS
port(
 inNumber :  IN wiredOrInt range 10 to 10 ;
 returnVar_main :  OUT wiredOrInt range 156232568 to 25 ;
 CLOCK : IN std_logic ;
 RESET : IN std_logic ;
 done : OUT std_logic );
END main;

ARCHITECTURE behav OF main IS
signal reminder : wiredOrInt range 104 to 138564864 ;
signal sumDivider : wiredOrInt range 156232568 to 33 ;
signal i : wiredOrInt range 156232568 to 32 ;
signal sT0_6 : wiredOrInt range 156232568 to 32 ;
signal sT1_6 : wiredOrBoolean  ;
signal sT2_8 : wiredOrBoolean  ;
signal sT3_15 : wiredOrBoolean  ;
```

```
BEGIN
   PROCESS
   BEGIN
   wait until CLOCK'event and CLOCK = '1';

   sumDivider  <=  0;
   i  <=  1;
while    sT0_6  <=  (inNumber / 2)
 loop
   reminder  <=  (inNumber rem i);
   sT2_8  <=  (reminder = 0);
   if sT2_8 then
    sumDivider  <=  (sumDivider + i);
   end if;
   i  <=  (i + 1);
end loop;
   sT3_15  <=  (sumDivider = inNumber);
   if sT3_15 then
    returnVar_main  <=  1;
       else
    returnVar_main  <=  0;
       end if;

 END PROCESS;
END behav;
```

SPARK je uspješno pretvorio zadanu C funkciju „main“ sa jednim ulaznim argumentom (number) i jednim izlaznim argumentom (vrijednost same funkcije) u entitet „main“ sa tri ulazna (number, CLOCK, RESET) te dva izlazna argumenta (returnVar i done). Razlog „viška“ argumenata je taj što je VHDL jezik za opisivanje digitalnih sklopova pa je SPARK dodao argumente/signale za upravljanje samim digitalnim sklopom koje slobodno možemo ignorirati u kontekstu prevođenja VHDL koda u QCL kod. Iako je SPARK uspješno preveo prethodno navedeni primjer C koda potrebno je napomenuti da postoje brojna ograničenja u upotrebi tog prevoditelja. Jedan od autora SPARK-a naveo je ograničenja prevoditelja u kontekstu da ne podržava pointere, rekurziju, goto naredbu, break i continue naredbe te višedimenzionalne redove [39]. Autori Arcilio J. Virginia, Yana D. Yankova, Koen L.M. Bertels su u svom radu opisali provedeno testiranje prevoditelja izvornoga koda C programskog jezika u VHDL kod [108]. U tim testovima SPARK je podržavao tek 54% funkcionalnosti C programskog jezika te je točnost SPARK-a bila 40% u testovima koje je prevoditelj uspio završiti. Dakle, treba imati na umu da će uspješnost prevođenja izvornih kodova drugih programskih jezika u QCL kod upotrebom postojećih prevoditelja drugih programskih jezika u VHDL ovisiti i o ograničenjima (npr. kvaliteta prevođenja, točnost

prevođenja, skup naredbi koje prevoditelj prepoznaje) tih postojećih prevoditelja koja proizlaze iz implementacije tih prevoditelja te kompleksnosti izvornog koda koji se prevodi.

6.2.2. Prevođenje VHDL izvornoga koda u reverzibilno sklopovlje upotrebom proširenih tablica istinitosti

Upravo iz razloga što VHDL ima logiku na sklopovskoj razini, vrlo jednostavno možemo prevoditi izvorni kod VHDL-a u tablice istinitosti upotrebom alata za testiranje i simuliranje VHDL koda. Jedan od takvih je GHDL [63], koji je u implementaciji iskorišten na način da na temelju svih mogućih ulaza u VHDL program generira sve moguće izlaze, te se na takav način kreira tablica istinitosti koja definira logiku VHDL programa.

Npr., uzmimo sljedeći izvorni VHDL kod programa koji ima tri ulazna i dva izlazna bita

```
library IEEE;
use IEEE.std_logic_1164.all;
entity TRUTH_TABLE_3_2_2 is
 port
 (
    a : in std_logic;
    b : in std_logic;
    c : in std_logic;
    d : out std_logic;
    e : out std_logic
 );
end entity TRUTH_TABLE_3_2_2;
architecture behaviour of TRUTH_TABLE_3_2_2 is
begin
  process(a,b,c)
  begin
    if (a = '0' and b = '0' and c = '0') then
      d <= '1';
      e <= '1';
    elsif (a = '0' and b = '0' and c = '1') then
      d <= '0';
      e <= '0';
    elsif (a = '0' and b = '1' and c = '0') then
      d <= '1';
      e <= '1';
    elsif (a = '0' and b = '1' and c = '1') then
      d <= '1';
      e <= '0';
    elsif (a = '1' and b = '0' and c = '0') then
      d <= '0';
      e <= '0';
```

```
   elsif (a = '1' and b = '0' and c = '1') then
     d <= '0';
     e <= '0';
   elsif (a = '1' and b = '1' and c = '0') then
     d <= '1';
     e <= '1';
   elsif (a = '1' and b = '1' and c = '1') then
     d <= '1';
     e <= '0';
   end if;
 end process;
end architecture behaviour;
```

Program za generiranje tablice istinitosti na temelju navedenog VHDL koda je

```
library IEEE;
use IEEE.std_logic_1164.all;
use std.textio.all; -- Imports the standard textio package.
    -- A testbench has no ports.
entity TRUTH_TABLE_3_2_2_tb is
end TRUTH_TABLE_3_2_2_tb;

architecture behav of TRUTH_TABLE_3_2_2_tb is
function bit_to_string(x : bit)
  return String is
  begin
  if (x = '1') then
     return "1";
   else
     return "0";
   end if;
end bit_to_string;
      -- Declaration of the component that will be instantiated.
component TRUTH_TABLE_3_2_2
        port (A : in STD_LOGIC; B : in STD_LOGIC; C : in STD_LOGIC; D : out STD_LOGIC; E : out
STD_LOGIC);
end component;
-- Specifies which entity is bound with the component.
 for TRUTH_TABLE_3_2_2_0: TRUTH_TABLE_3_2_2 use entity work.TRUTH_TABLE_3_2_2;
   signal A : STD_LOGIC;
   signal B : STD_LOGIC;
   signal C : STD_LOGIC;
   signal D : STD_LOGIC;
   signal E : STD_LOGIC;

 file l_file: TEXT open write_mode is
"/root/test/TRUTH_TABLE_3_2/TRUTH_TABLE_3_2_2/TRUTH_TABLE_3_2_2.pla";
    begin
      -- Component instantiation.
    TRUTH_TABLE_3_2_2_0: TRUTH_TABLE_3_2_2 port map (A => A, B => B, C => C, D => D, E => E);
    -- This process does the real job.
```

```
      process
type pattern_type is record
  -- Inputs and outputs
  A : STD_LOGIC;
  B : STD_LOGIC;
  C : STD_LOGIC;
end record;

variable my_line,line_file : line;
-- The patterns to apply.
type pattern_array is array (natural range <>) of pattern_type;
constant patterns : pattern_array :=
  (( '0', '0', '0' ),
   ( '0', '0', '1' ),
   ( '0', '1', '0' ),
   ( '0', '1', '1' ),
   ( '1', '0', '0' ),
   ( '1', '0', '1' ),
   ( '1', '1', '0' ),
   ( '1', '1', '1' ));

 begin
-- Check each pattern.
    for istep in patterns'range loop
     -- Set the inputs.
      A <= patterns(istep).A;
      B <= patterns(istep).B;
      C <= patterns(istep).C;

     -- Wait for the results.
      wait for 1 ns;
-- write to PLA file
      write(line_file, bit_to_string(to_bit(A))&bit_to_string(to_bit(B))&bit_to_string(to_bit(C))&"
            "&bit_to_string(to_bit(D))&bit_to_string(to_bit(E)));
      writeline(l_file, line_file);
    end loop;
    wait;
  end process;
 end behav;
```

Nakon što se prethodna skripta izvrši, rezultat je generirana PLA datoteka, čiji sadržaj je tablica istinitosti za VHDL program. Potrebno je napomenuti da VHDL program za generiranje tablice istinitosti na temelju VHDL koda prilagođen za podatkovni tip std_logic za ulaze i izlaze VHDL programa na temelju kojeg se generira tablica istinitosti, međutim sa malo prilagodbe moguće je jednostavno proširiti taj VHDL program da radi i sa ostalim tipovima podataka.

Tablica 10: Tablica istinitosti za VHDL program

Ulaz			**Izlaz**	
a	*b*	*c*	*d*	*e*
0	0	0	1	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

Ograničenje opisanog pristupa implementaciji generiranja tablice istinitosti je potrebno vrijeme za generiranje tablice istinitosti koje eksponencijalno ovisi o broju bitova kojima su zadani ulazni parametri u izvorni program te o složenosti samog programa. Ubrzo sama pretvorba izvornoga koda u tablicu istinitosti traje jako dugo a sama tablica sadrži jako puno redova što će u konačnici rezultirati jako dugim izvršavanjem sljedećih koraka prevođenja. Isto tako, kod viših programskih jezika za klasična računala (npr. Java, php, C++) spomenuta implementacija pretvorbe izvornog koda u tablicu istinitosti bit će realno izvedljiva na jednostavnijim slučajevima u kojima su eksplicitno definirani ulazni i izlazni parametri te su oni ujedno i jedini izvori i odredišta podataka programa. U praksi, programi pisani u višim programskim jezicima (npr. poslovna aplikacija) imaju jako puno različitih izvora i odredišta podataka (npr. baze podataka, web servisi, datoteke) a ne samo ulazne i izlazne parametre koje spomenuti implementacijski pristup uzima u obzir. Iz navedenog razloga spomenuta implementacija pretvorbe izvornoga koda viših programskih jezika u tablicu istinitosti je realno neizvediva u složenijim slučajevima te to predstavlja ograničenje implementacije predstavljenog pristupa za prevođenje izvornoga programskog koda klasičnoga računala u izvorni programski kod kvantnoga računala koji je predstavljen u poglavlju 6.1.

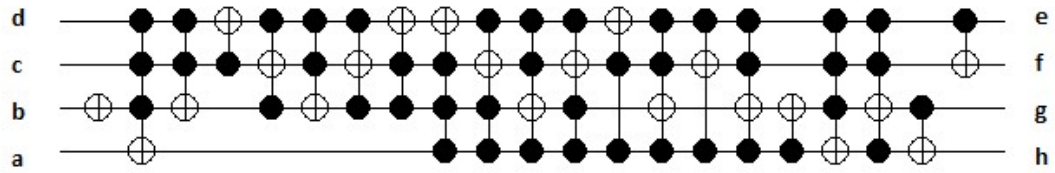
Nastavno na primjer prevođenja, proširivanje tablice istinitosti koja je navedena u prethodnoj tablici sam realizirao pomoću Revkit biblioteke [54], koja sadrži implementiran algoritam opisan u poglavlju 5.2.

Proširena tablica istinitosti za VHDL program izgleda ovako:

Tablica 11: Proširena tablica istinitosti za VHDL program

Ulaz				**Izlaz**			
a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
0	0	0	0	1	1	0	0
0	0	0	1	0	0	0	1
0	0	1	0	1	1	1	0
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	1	1	1	1
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	0	1	0
1	0	1	1	0	0	1	0
1	1	0	0	0	1	0	0
1	1	0	1	0	1	0	1
1	1	1	0	0	1	1	0
1	1	1	1	0	1	1	1

Reverzibilno sklopovlje sam generirao na temelju proširene tablice istinitosti upotrebom Revkit biblioteke, koja ima implementiran Millerov, Maslov i Dueckov algoritam opisan u poglavlju 6.1.1. Generirano reverzibilno sklopovlje na temelju proširene tablice istinitosti za VHDL program je prikazano na sljedećoj slici.

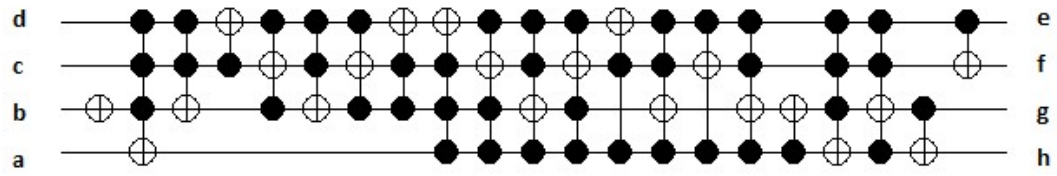


Slika 41: Generirano reverzibilno sklopovlje na temelju proširene tablice istinitosti

6.2.3. Prevođenje reverzibilnog sklopovlja u kvantno sklopovlje

Implementirani prevoditelj ima mogućnost korištenja jednog od tri algoritma za generiranje kvantnog sklopovlja na temelju reverzibilnog sklopovlja. Prvi algoritam izravno

pretvara reverzibilno sklopovlja u kvantno, jer je reverzibilno sklopovlje generirano na temelju proširene tablice istinitosti upotrebom Millerovog, Maslovog i Dueckovog algoritma (poglavlje 6.1.1.), koji osigurava da generirana reverzibilna sklopovlja imaju svoju izravnu reprezentaciju u kvantnom sklopovlju. Drugi algoritam je CS dekompozicija koja je opisana u poglavlju 6.1.2.1., a treći algoritam je QR dekompozicija koja je opisana u poglavlju 6.1.2.2. Za izvršavanje algoritama CS i QR dekompozicija koristio sam LAPACK biblioteku [56]. Važno je napomenuti da je generirano kvantno sklopovlje u ovom koraku svedeno na kanonski oblik u smislu da su sva kontrolirana vrata viših razina (osim C^nNOT) dekomponirana na način kako je to prikazano na slici 12 u poglavlju 5.8.2.2. Glavni razlog za svođenje na kanonski oblik je ograničenje QCL jezika, u kojem se ne može na jednostavan način definirati proizvoljna kontrolirana vrata viših razina. Generirano kvantno sklopovlje na temelju reverzibilnog sklopovlja sa slike 40 upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno je identično reverzibilnom sklopovlju sa slike 40.



Slika 42: Generirano kvantno sklopovlje na temelju reverzibilnog sklopovlja sa slike 40 upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno

6.2.4. Prevođenje kvantnog sklopovlja u QCL izvorni kod

Quantum Computer Language (QCL) je jedan od najnaprednijih programskih jezika za kvantna računala. On ima funkcionalnost kvantnoga simulatora te koristi sintaksu koja je izvedena iz sintakse programskog jezika C. Vrlo je korisnički pristupačan te uključuje neke funkcionalnosti poput korisnički definiranih funkcija i operatora, koje su karakteristične za više programske jezike [78][98]. Logika QCL programa je također na razini sklopovlja, što nam omogućuje da vrlo lako napravimo pretvaranje kvantnih vrata iz kvantnog sklopovlja u konkretne naredbe u izvornom kodu QCL-a. Dakle, završni korak u implementaciji prevoditelja je pretvaranje kvantnih vrata iz kvantnih sklopova u izvršni kod QCL programskog jezika. Prevedeni VHDL kod za kvantno sklopovlje sa slike 41 izgleda ovako:

```
string bitToString(int q, int bit) { // funkcija za prilagođavanje ispisa
```

```
 if bit(q, bit){
 return "1";
 }else{
 return "0";
            }
}
 qufunct init(qureg q, int state) { // funkcija za inicijalizaciju ulaznih kvantnih bitova
if bit(state,0){
 Not(q[3]);
}
if bit(state,1){
 Not(q[2]);
}
if bit(state,2){
 Not(q[1]);
}
}
procedure process(qureg q) { // implementacija kvantnoga sklopa sa slike 41
Not(q[1]); // primjena Not kvantnih vrata na drugi kvantni bit
CNot(q[0], q[1]&q[2]&q[3]); // primjena C^nNot kvantnih vrata na prvi kvantni bit u ovisnosti o
drugom, trećem i četvrtom kvantnom bitu
CNot(q[1], q[2]&q[3]);
CNot(q[3], q[2]);
CNot(q[2], q[1]&q[3]);
CNot(q[1], q[2]&q[3]);
CNot(q[2], q[1]&q[3]);
CNot(q[3], q[1]&q[2]);
CNot(q[3], q[0]&q[1]&q[2]);
CNot(q[2], q[0]&q[1]&q[3]);
CNot(q[1], q[0]&q[2]&q[3]);
CNot(q[2], q[0]&q[1]&q[3]);
CNot(q[3], q[0]&q[2]);
CNot(q[1], q[0]&q[2]&q[3]);
CNot(q[2], q[0]&q[3]);
CNot(q[1], q[0]&q[2]&q[3]);
CNot(q[1], q[0]);
CNot(q[0], q[1]&q[2]&q[3]);
CNot(q[1], q[0]&q[2]&q[3]);
CNot(q[0], q[1]);
CNot(q[2], q[3]);
}
 qureg q[4]; // inicijalizacija kvantnoga registra od 4 kvantna bita.
 int x;
 int i;
print "Broj kvantnih bitova: 4";
print "Ulazni kvantni bitovi: 2,1,0"; // ispis ulaznih kvantnih bitova
print "Izlazni kvantni bitovi: 0,1"; // ispis izlaznih kvantnih bitova
 for i=0 to 7 step 1 // petlja koja će ispitivati sve binarne ulaze u program
 {
 reset; // postavljanje kvantnoga registra u stanje |000>
 init(q,i); // poziv funkcije koja će inicijalizirati ulazne kvantne bitove u ovisnosti o iteraciji petlje
 process(q); // procesiranje, primjena kvantnih vratiju na kvantne bitove registra
```

```
dump;  // ispis vjerojatnosnih amplituda pojedinih izlaza
measure q,x; // mjerenje kvantnoga registra
print bitToString(i,2)&bitToString(i,1)&bitToString(i,0)&bitToString(x,0)&bitToString(x,1); //
ispisivanje rezultata kvantnoga registra u obliku tablice istinitosti
}
```

Izlaz iz generiranog QCL koda su vjerojatnosti vrijednosti izlaznih kvantnih bitova, npr. za izlaze *e* i *f* u prvom retku tablice istinitosti to su vrijednosti 1|1> i 1|1> te sama tablica istinitosti koja je generirana na temelju mjerenja kvantnog registra nakon izvođenja QCL koda za pojedini ulaz.

Tablica 12: Tablica istinitosti QCL programa

Ulaz			**Izlaz**	
b	*c*	*d*	*e*	*f*
0	0	0	1	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

7. Testiranje generiranja kvantnog sklopovlja upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, te upotrebom QR i CS algoritama

U ovom dijelu opisat ću proces testiranja koji sam proveo kako bih saznao kakav je odnos između algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, CS algoritma i QR algoritma po pitanju kvantnoga troška i brzine generiranja kvantnih sklopova nad skupom podataka koji predstavlja programe klasičnih računala. Navest ću rezultate testiranja, te zaključak temeljen na istima.

7.1. Izvor podataka

Kao izvor podataka koristio sam izvorne VHDL datoteke generirane generatorom izvornoga VHDL koda kojeg generira izvorni VHDL kod za digitalni sklop koji implementira nasumičnu tablicu istinitosti za zadani broj ulaznih i izlaznih bitova. Za generator izvornoga VHDL koda sam se odlučio iz dva razloga. Prvi je taj što smatram da će takav generator omogućiti reprezentativniji uzorak za testiranje implementiranog prevoditelja u smislu općenitosti, zbog toga što se generirani uzorak izvornoga VHDL koda ne temelji na rješenjima za neke konkretne probleme već je generiran na temelju nasumičnih tablica istinitosti. Drugi razlog je sama količina izvornih VHDL programa s kojima sam provodio testiranje implementiranog prevoditelja. Testiranje implementiranog prevoditelja sam proveo nad 3900 VHDL programa, te bi takav uzorak izvornih VHDL datoteka bilo gotovo nemoguće sastaviti ako bi se svi implementirani VHDL programi u uzorku temeljili na tablicama istinitosti za neki konkretni problem. Generator izvornoga koda VHDL datoteka implementiran je u sklopu provođenja ovog istraživanja u programskom jeziku C++, a sve generirane VHDL datoteke koje su bile korištene kao izvor podataka nalaze se na CD-u koji je priložen uz ovaj rad.

7.2. Opis procesa testiranja

Izvorne VHDL datoteke su preko 2. i 3. koraka sa slike 39 pretvorene u kvantno sklopovlje, pri čemu je reverzibilno sklopovlje dobiveno nakon 2. koraka izravno pretvarano

u kvantno sklopovlje, te je izračunata unitarna matrica takvog sklopovlja. Ta unitarna matrica je zapravo reprezentacija logike klasičnoga programa u logici kvantnoga računala. Unitarna matrica potom je dekomponirana upotrebom QR i CS dekompozicijskih algoritama, te je na temelju dobivenih faktora dekompozicije sastavljeno kvantno sklopovlje. Prilikom samog procesa testiranja mjerio sam zasebno vrijeme potrebno za generiranje kvantnog sklopovlja upotrebom spomenutih algoritama na temelju reverzibilnog sklopovlja (algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno), odnosno unitarne matrice (QR i CS algoritmi). Također sam mjerio i kvantni trošak generiranih kvantnih sklopova te duljinu izvornoga reverzibilnog sklopovlja. Testiranje je provedeno nad reverzibilnim sklopovima s 1, 2, 3, 4, 5, 6, 7, 8, 9 i 10 bitova, odnosno unitarnim matricama dimenzija $2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16, 32 \times 32, 64 \times 64, 128 \times 128, 256 \times 256, 512 \times 512$ i 1024×1024. Testiranje je provedeno na računalu s 4 GB RAM-a i i5 2.3 GHz-nim procesorom na Linux Ubuntu 13.10 operacijskom sustavu.

7.3. Rezultati testiranja

Tablica 13: Korištene oznake

Oznaka	Značenje
rb_n	Broj bitova u reverzibilnom sklopovlju
N_{test}	Broj provedenih testova
$N_{\leq rqc}$	Broj testova u kojima je kvantni trošak generiranog kvantnog sklopovlja manji ili jednak kvantnom trošku reverzibilnog sklopovlja
$N_{>rqc}$	Broj testova u kojima je kvantni trošak generiranog kvantnog sklopovlja veći od kvantnoga troška reverzibilnog sklopovlja
rcl_{min}	Minimalna duljina reverzibilnog sklopovlja
rcl_{max}	Maksimalna duljina reverzibilnog sklopovlja
rcl_{avg}	Prosječna duljina reverzibilnog sklopovlja
t_{min} (µs)	Minimalno vrijeme potrebno za generiranje kvantnog sklopovlja u mikrosekundama
t_{max} (µs)	Maksimalno vrijeme potrebno za generiranje kvantnog sklopovlja u mikrosekundama
t_{avg} (µs)	Prosječno vrijeme potrebno za generiranje kvantnog sklopovlja u mikrosekundama
qc_{min}	Minimalni kvantni trošak
qc_{max}	Maksimalni kvantni trošak
qc_{avg}	Prosječni kvantni trošak

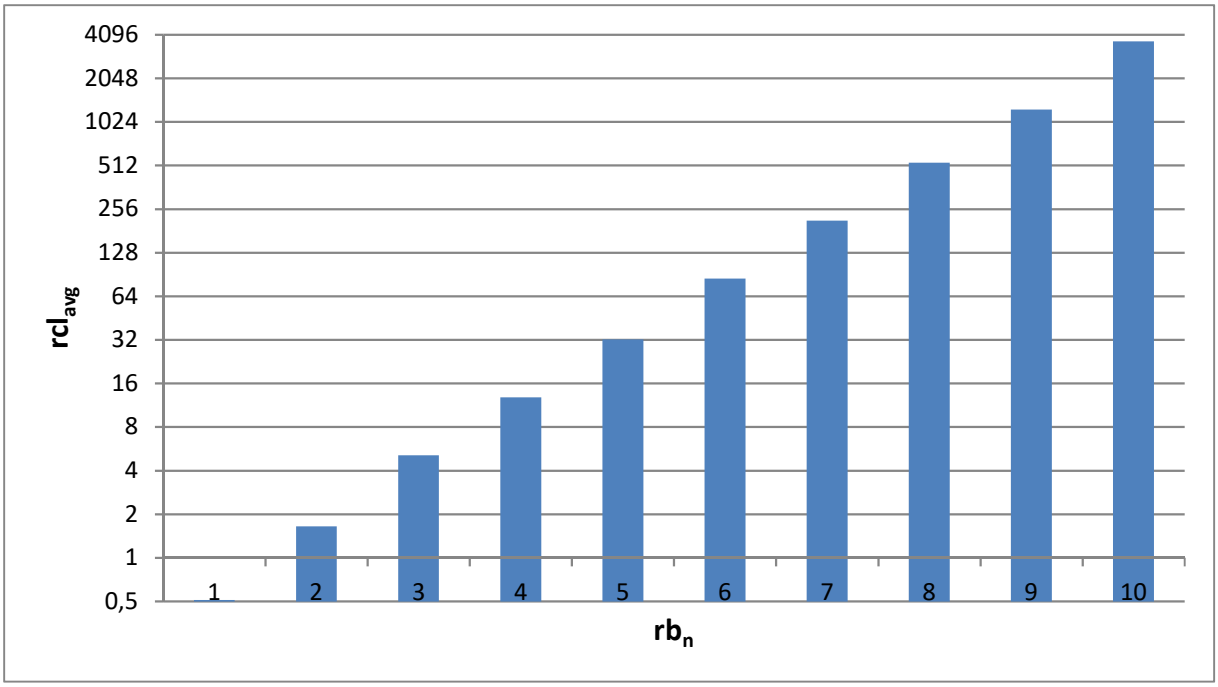
7.3.1. Algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno

Tablica 14: Rezultati testiranja generiranja kvantnog sklopovlja upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno

rb_n	N_{test}	$N_{\leq rqc}$	$N_{>rqc}$	rcl_{min}	rcl_{max}	rcl_{avg}	t_{min}(µs)	t_{max} (µs)	t_{avg}(µs)	qc_{min}	qc_{max}	qc_{avg}
1	49	49	0	0	0	0,51	16	74	24,72	0	1	0,51
2	176	176	0	0	4	1,65	17	93	39,02	0	4	1,65
3	332	332	0	0	12	5,09	20	277	64,7	0	40	13,83
4	373	373	0	1	33	12,83	29	918	112,48	4	297	91,98
5	480	480	0	3	68	32,19	32	1570	238,73	23	1620	580,25
6	564	564	0	8	150	84,46	63	2133	611,6	91	7750	3734,51
7	519	519	0	23	359	212,27	34	5168	1389,89	660	40411	22481,03
8	639	639	0	50	804	535,79	81	7183	3851,6	3154	213410	133388,5
9	513	513	0	108	1841	1245,61	824	18719	10173,31	12274	1047803	713326,45
10	255	255	0	2359	4126	3671,93	20887	37895	30851,55	3611979	5407211	4816615,42

Varijabilni broj testova za pojedinu testnu skupinu je posljedica nasumičnog generiranja VHDL izvornih kodova na temelju kojih se izvodila proširiva tablica istinitosti, a tako i samo reverzibilno sklopovlje. Tako će, primjerice, u jednom slučaju VHDL program koji ima tri ulazna i dva izlazna bita rezultirati reverzibilnim sklopovljem od četiri bita, dok će u drugom slučaju drugi VHDL program s istim brojem ulaznih i izlaznih bitova rezultirati reverzibilnim sklopovljem od tri bita.

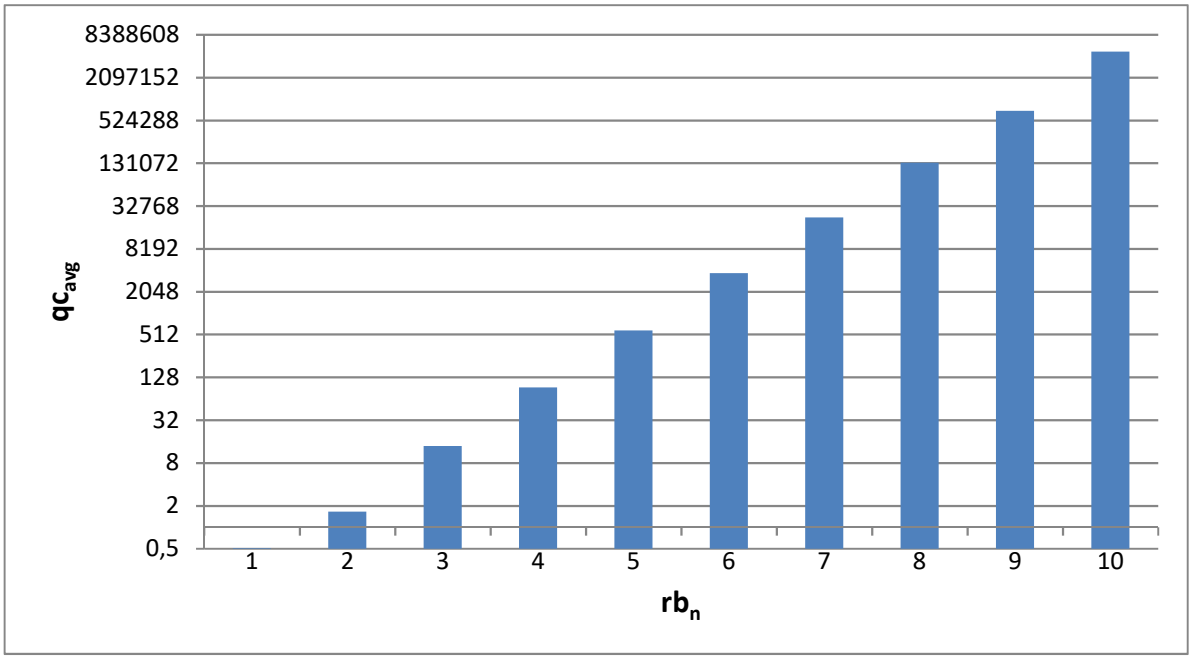
Sljedeći grafikon prikazuje prosječnu duljinu reverzibilnog sklopovlja u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.



Grafikon 1: Prosječna duljina reverzibilnog sklopovlja u odnosu na broj bitova u reverzibilnom sklopovlju (algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno)

Na prethodnom grafikonu lako je uočljiva eksponencijalna ovisnost duljine reverzibilnog sklopovlja u odnosu na broj bitova reverzibilnog sklopovlja. Razlog tome je Millerov, Maslov i Dueckov algoritam, koji generira reverzibilno sklopovlje na temelju proširive tablice istinitosti čiji broj i složenost redaka ovise o broju ulaznih i izlaznih bitova VHDL programa na temelju kojih je generirana izvorna tablica istinitosti. Ta veza između broja ulaznih i izlaznih bitova VHDL programa i broja bitova reverzibilnog sklopovlja bit će detaljnije objašnjena u poglavlju 8.3.1..

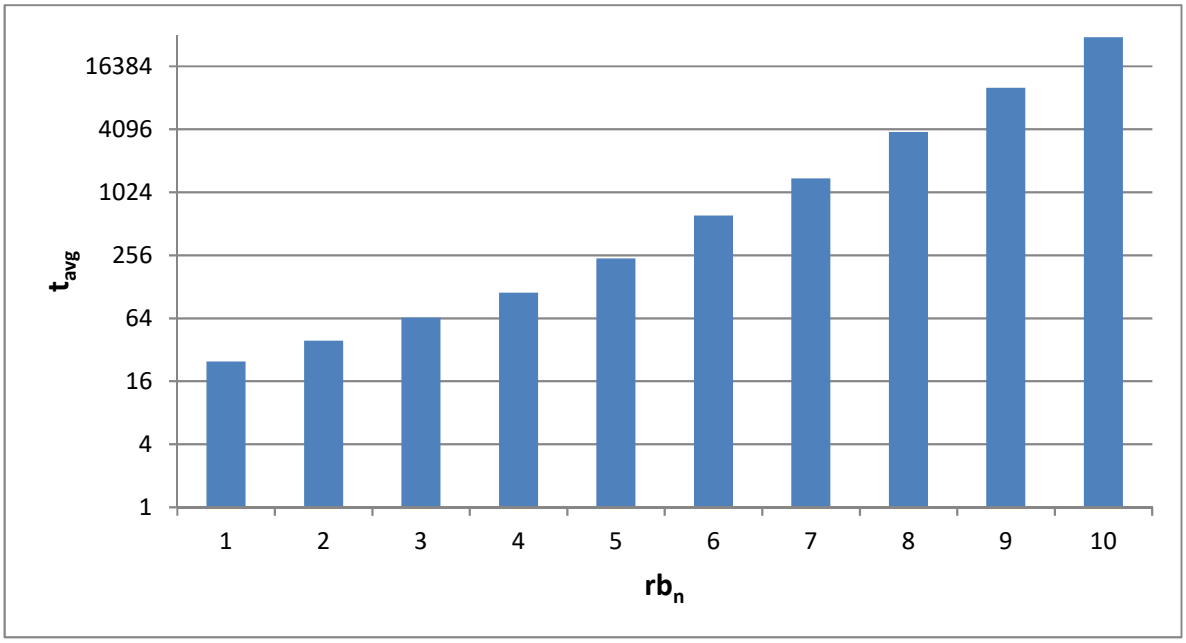
Sljedeći grafikon prikazuje prosječni kvantni trošak generiranih kvantnih sklopova u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.



Grafikon 2: Prosječni kvantni trošak kvantnog sklopovlja u odnosu na broj bitova u reverzibilnom sklopovlju (algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno)

Iz razloga što algoritam izravno pretvara reverzibilna vrata u kvanta, kvantni trošak se pri toj pretvorbi ne mijenja. Kvantni trošak je polinomno ovisan o duljini reverzibilnog sklopovlja, što je vidljivo na prethodnom grafikonu koji ima sličan trend kao i grafikon 1.

Sljedeći grafikon prikazuje prosječno vrijeme izvođenja provedenih testova upotrebom algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno u mikrosekundama u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.



Grafikon 3: Prosječno vrijeme trajanja testova u odnosu na broj bitova u reverzibilnom sklopovlju (algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno)

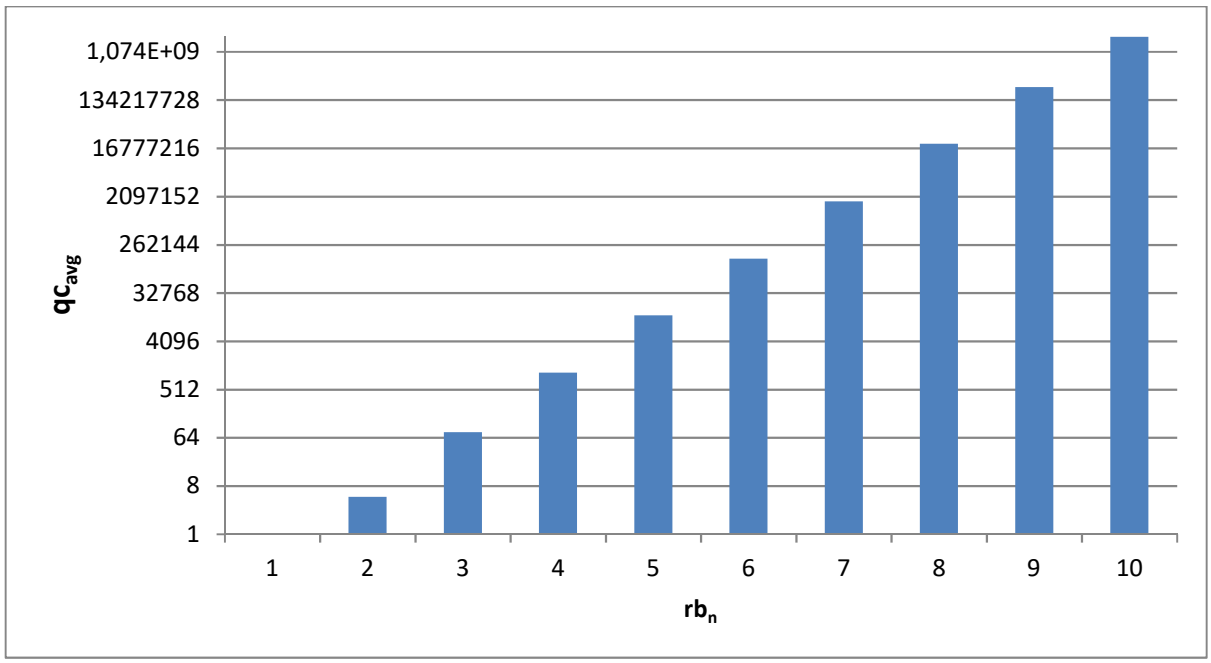
S obzirom da je reverzibilno sklopovlje sastavljeno od reverzibilnih vrata koja imaju direktnu reprezentaciju u kvantnom sklopovlju, kod pretvaranja reverzibilnog sklopovlja u kvantno potrebno je sva reverzibilna vrata pretvoriti u kvantna vrata, pa zbog toga vremensko trajanje testova u odnosu na broj bitova u reverzibilnom sklopovlju polinomno ovisi o duljini reverzibilnog sklopovlja, odnosno eksponencijalno o broju bitova reverzibilnog sklopovlja što je i uočljivo na prethodnom grafikonu.

7.3.2. QR dekompozicija

Tablica 15: Rezultati testiranja generiranja kvantnog sklopovlja upotrebom QR dekompozicije

rb_n	N_{test}	$N_{\leq rqc}$	$N_{>rqc}$	t_{min} (µs)	t_{max} (µs)	t_{avg}(µs)	qc_{min}	qc_{max}	qc_{avg}
1	49	23	26	18	120	34,02	0	2	1,02
2	176	77	99	23	890	122,73	0	10	4,97
3	332	20	312	54	1644	356,18	0	174	80,84
4	373	0	373	269	3880	1613,78	34	2286	1054,27
5	480	0	480	5303	57902	16322,73	2420	24281	12526,03
6	564	0	564	91092	338843	205535,7	60116	245947	143710,3
7	519	0	519	1172931	4560456	2871267,37	652900	2710438	1693631,61
8	639	0	639	17019220	71025806	41139864,95	8830632	28380262	20256996,82
9	513	0	513	245302730	904321165	587412150,4	103325955	478444158	234731708,7
10	255	0	255	14086671	6421089382	3510634911	1264785674	2912550306	2061162206

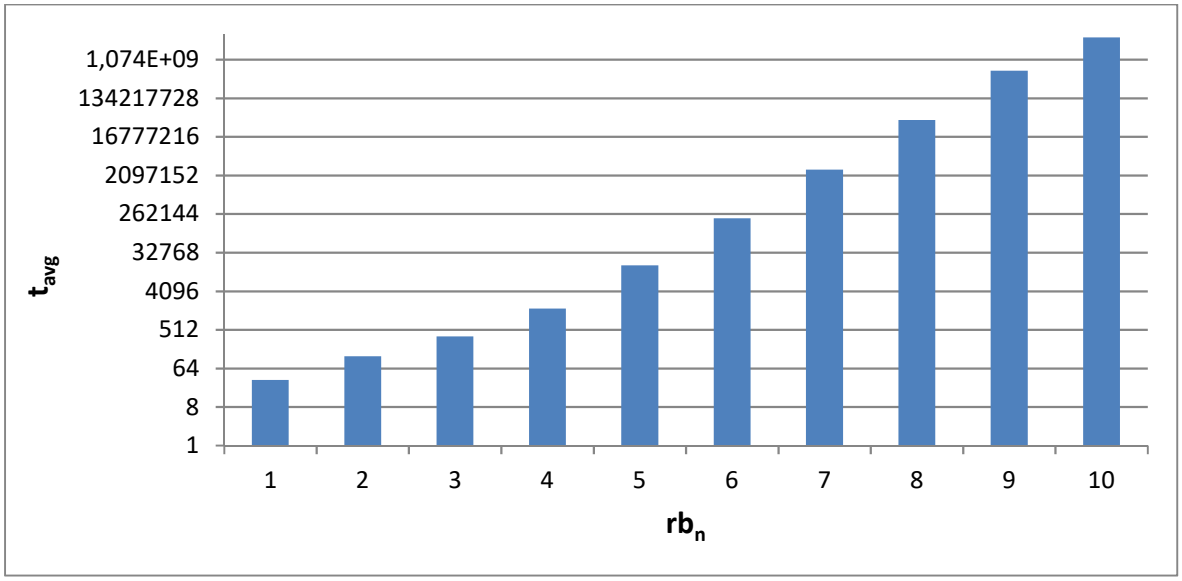
Sljedeći grafikon prikazuje prosječni kvantni trošak generiranih kvantnih sklopova upotrebom QR dekompozicije u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.



Grafikon 4: Prosječni kvantni trošak kvantnog sklopovlja u odnosu na broj bitova u reverzibilnom sklopovlju (QR dekompozicija)

Kako broj elemenata u unitarnoj matrici koja se dekomponira QR algoritmom ovisi eksponencijalno o broju bitova reverzibilnog sklopovlja, tako i broj Givensonovih rotacija koji se temelji na elementima unitarne matrice smještenim ispod glavne dijagonale eksponencijalno ovisi o broju bitova reverzibilnog sklopovlja. S obzirom da za svaku Givensonovu rotaciju u generiranom kvantnom sklopovlju nastaju jedna kontrolna kvantna vrata višeg reda, a ukupni kvantni trošak generiranog kvantnog sklopovlja ovisi o broju kvantnih vrata te o kvantnom trošku pojedinačnih kvantnih vrata, to u konačnici rezultira eksponencijalnom ovisnosti kvantnoga troška generiranog kvantnog sklopovlja o broju bitova reverzibilnog sklopovlja, što je i lako uočljivo na prethodnom grafikonu. Također, iz kolone $N_{\leq rqc}$ u tablici 12 je vidljivo je da broj testova u kojima je kvantni trošak generiranog kvantnog sklopovlja manji ili jednak kvantnom trošku reverzibilnog sklopovlja značajno pada s količinom reverzibilnih bitova, odnosno dimenzijom unitarne matrice.

Sljedeći grafikon prikazuje prosječno vrijeme izvođenja testova upotrebom QR dekompozicije u mikrosekundama u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.



Grafikon 5: Prosječno vrijeme trajanja testova u odnosu na broj bitova u reverzibilnom sklopovlju (QR dekompozicija)

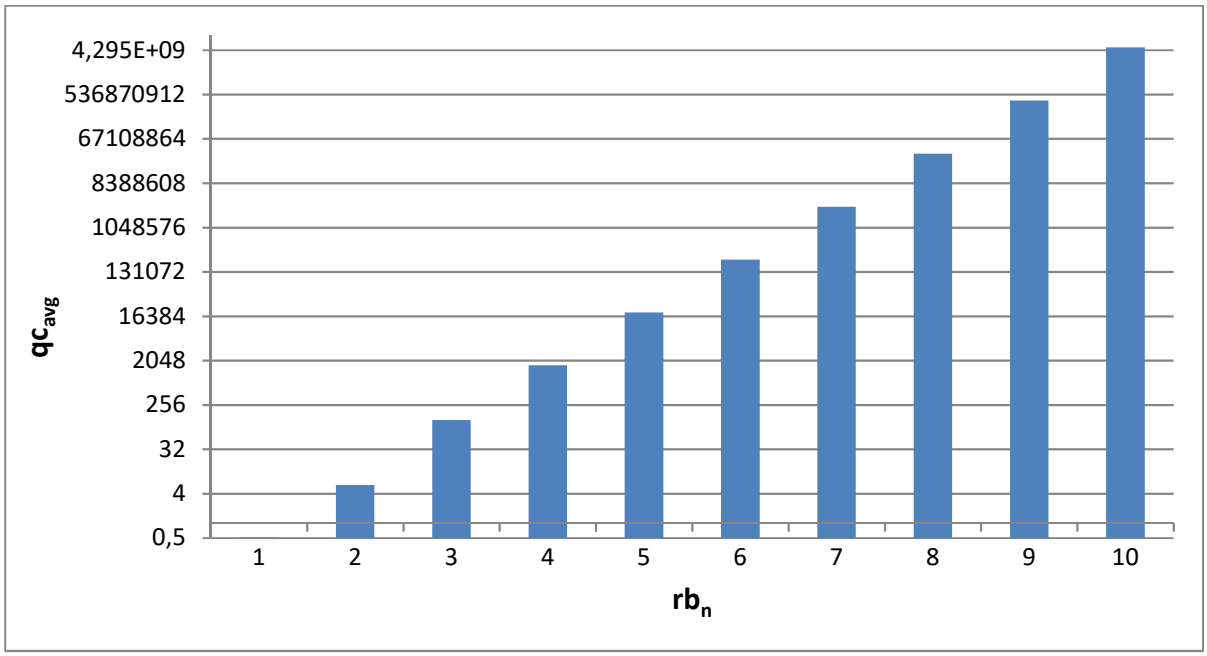
Kao kod grupe testova u kojima je korišten algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, tako je i kod grupe testova u kojima je bio korišten algoritam QR dekompozicije uočljiva eksponencijalna ovisnost trajanja izvođenja grupe testova u ovisnosti o broju bitova reverzibilnog sklopovlja na temelju kojeg se generira kvantno sklopovlje. S obzirom da broj bitova reverzibilnog sklopovlja utječe eksponencijalno na dimenziju matrice koja se dekomponira, a QR algoritam radi Givensonove rotacije na temelju elemenata matrice ispod glavne dijagonale, razumljiva je eksponencijalna ovisnost prosječnog vremena izvođenja testova i broja bitova reverzibilnog sklopovlja.

7.3.3. CS dekompozicija

Tablica 16: Rezultati testiranja generiranja kvantnog sklopovlja upotrebom CS dekompozicije

rb_n	N_{test}	$N_{\leq rqc}$	$N_{>rqc}$	t_{min} (µs)	t_{max} (µs)	t_{avg}(µs)	qc_{min}	qc_{max}	qc_{avg}
1	49	49	0	15	75	20,81	0	1	0,51
2	176	41	135	17	282	114,7	0	12	5,99
3	332	11	321	18	1567	431	0	196	125,52
4	373	0	373	286	3917	1465,52	289	2318	1630,71
5	480	0	480	2049	17169	5469,78	6069	24117	19623,14
6	564	0	564	11717	64254	25458,66	106262	262969	231327,39
7	519	0	519	86751	133558	111750,11	2153214	2986332	2778257,33
8	639	0	639	361393	1376088	503751,87	23976124	35145063	33470613,18
9	513	0	513	1814778	2398578	2162484,02	337132961	417367213	402903760,9
10	255	0	255	9373277	11632308	9779700,67	4709630442	4955781120	4886423661

Sljedeći grafikon prikazuje prosječni kvantni trošak generiranog sklopovlja upotrebom CS dekompozicije u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.

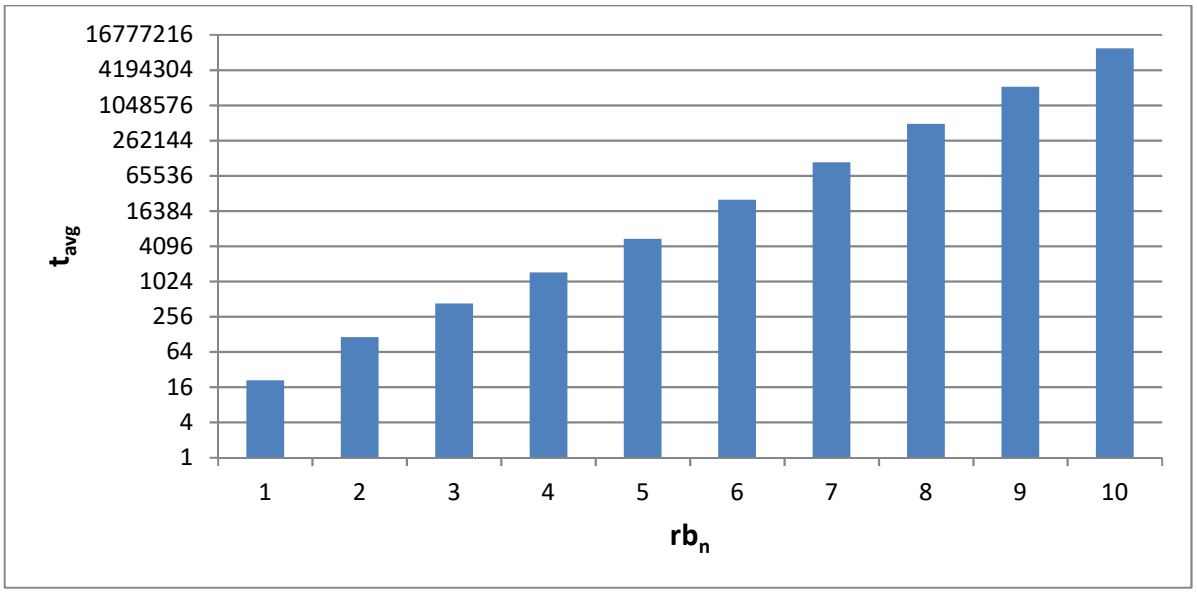


Grafikon 6: Prosječni kvantni trošak kvantnog sklopovlja u odnosu na broj bitova u reverzibilnom sklopovlju (CS dekompozicija)

CS algoritam je rekurzivan i tijekom izvođenja generira binarno stablo koje je prikazano na slici 37. Na određenoj razini generiranog binarnog stabla nalaze se unitarne matrice koje imaju dvostruko manju dimenziju od onih na prethodnoj razini. Dakle, binarno

stablo CS algoritma ima $(\log_2 N) - 1$ razina, gdje je N dimenzija inicijalne unitarne matrice koja se dekomponira. Broj faktora odnosno matrica koje se pretvaraju u kvantna vrata (listovi CS stabla i centralne matrice) eksponencijalno ovisi o broju bitova reverzibilnog sklopovlja na temelju kojeg se računa inicijalna unitarna matrica koja se dekomponira. Sve matrice CS binarnog stabla koje se pretvaraju u kvantna vrata se pretvaraju u kontrolirana kvantna vrata viših razina stoga je i razumljiva eksponencijalna ovisnost kvantnoga troška o broju bitova reverzibilnog sklopovlja koja je prikazana na prethodnom grafikonu. Slično kao i kod QR algoritma, iz kolone $N_{\leq rq}$ u tablici 13. je vidljivo da broj testova u kojima je kvantni trošak generiranog kvantnog sklopovlja manji ili jednak kvantnom trošku reverzibilnog sklopovlja značajno pada sa količinom reverzibilnih bitova, odnosno dimenzijom unitarne matrice.

Sljedeći grafikon prikazuje prosječno vrijeme trajanja testova upotrebom CS dekompozicije u mikrosekundama u odnosu na broj bitova u reverzibilnom sklopovlju na logaritamskoj skali.



Grafikon 7: Prosječno vrijeme trajanja testova u odnosu na broj bitova u reverzibilnom sklopovlju (CS dekompozicija)

Kao i kod prethodnih algoritama, tako i kod grupe testova u kojima je bila korištena CS dekompozicija primjetna je eksponencijalna vremenska složenost u odnosu na broj bitova reverzibilnog sklopovlja na temelju kojeg se računala unitarna matrica koja se dekomponirala. Kao što je već napomenuto, CS algoritam generira binarno stablo tijekom svog izvođenja. To binarno stablo ima $\log_4 N$ razina, odnosno broj faktora, matrica (listovi CS stabla i centralne

matrice) koje se pretvaraju u kvantna vrata, eksponencijalno će ovisiti o broju bitova reverzibilnog sklopovlja na temelju kojeg se računa inicijalna unitarna matrica koja se dekomponira. Stoga je i razumljiva eksponencijalna ovisnost prosječnog vremena izvođenja pojedine grupe testova o broju bitova reverzibilnog sklopovlja na temelju kojeg se računa inicijalna unitarna matrica.

7.4. Kritički osvrt

Sljedeća tablica prikazuje prosječna vremena izvođenja pojedinih grupa testova, te prosječni kvantni trošak pojedine grupe testova za svaki algoritam. Posebno su iskazani odnosi između prosječnih vremena i prosječnih kvantnih troškova kvantnih sklopova koji su bili generirani upotrebom CS i QR algoritama.

Tablica 17: Prosječna vremena trajanja, prosječni kvantni trošak i broj testova u kojima je kvantni trošak manji ili jednak kvantnom trošku reverzibilnog sklopovlja za svaki algoritam

rb_n	N_{test}	IP $N_{\leq rqc}$	QR $N_{\leq rqc}$	CS $N_{\leq rqc}$	IP $t_{avg}(\mu s)$	QR $t_{avg}(\mu s)$	CS $t_{avg}(\mu s)$	IP qc_{avg}	QR qc_{avg}	CS qc_{avg}	$\frac{CS\ t_{avg}(\mu s)}{QR\ t_{avg}(\mu s)}$	$\frac{CS\ qc_{avg}(\mu s)}{QR\ qc_{avg}(\mu s)}$
1	49	49	23	49	24,72	34,02	20,81	0,51	1,02	0,51	0,611699	0,5
2	176	176	77	41	39,02	122,73	114,7	1,65	4,97	5,99	0,934572	1,205231
3	332	332	20	11	64,7	356,18	431	13,83	80,84	125,52	1,210062	1,552697
4	373	373	0	0	112,48	1613,78	1465,52	91,98	1054,27	1630,71	0,908129	1,546767
5	480	480	0	0	238,73	16322,73	5469,78	580,25	12526,03	19623,14	0,335102	1,566589
6	564	564	0	0	611,6	205535,7	25458,66	3734,51	143710,3	231327,39	0,123865	1,609679
7	519	519	0	0	1389,89	2871267,37	111750,11	22481,03	1693631,61	2778257,33	0,03892	1,640414
8	639	639	0	0	3851,6	41139864,95	503751,87	133388,5	20256996,82	33470613,18	0,012245	1,652299
9	513	513	0	0	10173,31	587412150,4	2162484,02	713326,45	234731708,7	402903760,9	0,003681	1,716444
10	255	255	0	0	30851,55	3510634911	9779700,67	4816615,42	2061162206	4886423661	0,002786	2,370713

Što se tiče prosječnog vremena trajanja, grupe testova u kojima je korišten izravni algoritam za pretvorbu reverzibilnog sklopovlja u kvantno (IP) imaju najmanje prosječno trajanje, dok su se grupe testova u kojima je korišten CS algoritam znatno brže izvodile od onih u kojima je korišten QR algoritam. Za primijetiti je da se s porastom dimenzija unitarne matrice koja se dekomponira smanjuje omjer prosječnog vremena trajanja testova koji koriste CS algoritam i prosječnog vremena trajanja testova koji koriste QR algoritam. To znači da dimenzija unitarne matrice odnosno broj bitova reverzibilnog sklopovlja više utječe na povećanje prosječnog vremena trajanja QR algoritma nego što utječe na povećanje prosječnog vremena trajanja CS algoritma.

Najmanji prosječni kvantni trošak imaju kvantni sklopovi koji su generirani u testovima u kojima se koristi izravni algoritam za pretvorbu reverzibilnog sklopovlja u kvantno (IP), dok je prosječni kvantni trošak kvantnih sklopova koji su generirani u testovima koji koriste CS algoritam veći od prosječnog kvantnoga troška kvantnih sklopova koji su generirani u testovima u kojima se koristi QR algoritam. Isto tako, uočljivo je da omjer prosječnog kvantnoga troška kvantnih sklopova koji su generirani u testovima koji koriste CS algoritam i prosječnog kvantnoga troška kvantnih sklopova koji su generirani u testovima koji koriste QR algoritam raste s porastom dimenzije unitarne matrice koja se dekomponira. Na temelju toga možemo zaključiti da dimenzija unitarne matrice, na temelju koje se generira kvantno sklopovlje, ima veći utjecaj na povećanje kvantnoga troška kod onih kvantnih sklopova koji su generirani sa CS algoritmom nego kod onih koji su generirani s QR algoritmom.

Što se tiče broja testova u kojima je kvantni trošak generiranog sklopovlja manji ili jednak kvantnom trošku reverzibilnog sklopovlja vidimo da u slučaju QR i CS algoritama taj broj drastično pada s brojem bitova reverzibilnog sklopovlja, odnosno dimenzijom unitarne matrice. U slučaju algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, kvantni trošak generiranog kvantnog sklopovlja je uvijek jednak kvantnom trošku reverzibilnog sklopovlja.

8. Testiranje implementiranog prevoditelja izvornoga koda klasičnih računala u izvorni kod kvantnih računala

U ovom dijelu opisat ću proces testiranja implementiranog prevoditelja izvornoga VHDL koda u izvorni kod QCL-a kako bih saznao prosječnu točnost same pretvorbe. Prilikom prevođenja izvornoga VHDL koda u izvorni QCL kod koristio sam već spomenute algoritme za generiranje kvantnog sklopovlja - izravni algoritam za pretvaranje reverzibilnog sklopovlja u kvantno, CS algoritam i QR algoritam. Navest ću rezultate testiranja, te zaključak koji se na njima temelji.

8.1. Izvor podataka

Prilikom testiranja implementiranog prevoditelja koristio sam iste izvorne VHDL datoteke kao i u testiranju koje je opisano u poglavlju 7. Izvorne VHDL datoteke su generirane implementiranim generatorom za generiranje izvornih VHDL datoteka. Sve izvorne VHDL datoteke koje su korištene kao izvor podataka nalaze se na CD-u koji je priložen uz ovaj rad.

8.2. Opis procesa testiranja

Izvorni kod VHDL datoteka je na temelju opisanih koraka implementacije modela (slika 39) preveden u izvorni kod QCL jezika za kvantna računala. Prevođenje izvornoga koda napravljeno je zasebno upotrebom izravnog algoritma za pretvaranje reverzibilnog sklopovlja u kvantno, te zasebno upotrebom QR i CS dekompozicijskih algoritama za unitarne matrice. Prilikom samog procesa testiranja mjerio sam vrijeme potrebno za prevođenje izvornoga VHDL koda u QCL kod, duljinu reverzibilnog sklopovlja, broj bitova reverzibilnog sklopovlja, duljinu kvantnog sklopovlja, kvantni trošak generiranog sklopovlja, te ispravnost generiranog sklopovlja. Točnost generiranog QCL koda mjerio sam preko vjerojatnosti točnih izlaza koje će se izračunati na temelju elemenata vektora stanja kvantnoga sustava unutar QCL simulatora u trenutku nakon izvršavanja generiranog QCL izvornoga koda, a prije samog mjerenja izlaza za zadani ulaz.

Točnost pojedinog testa računao sam kao

$$Acc_u = p_i * 100,$$

gdje je

p_i - vjerojatnost točnosti izlaza *i* za zadani ulaz *u*

Prosječnu točnost pojedine grupe testova računao sam kao

$$Acc_{avg} = \frac{\left(\sum_{u=1}^{N_{test}} Acc_u\right)}{N_{test}},$$

gdje su

Acc_u - točnost pojedinog u-tog testa u grupi

N_{test}- ukupan broj testova u grupi

8.3. Rezultati testiranja

Tablica 18: Korištene oznake

Oznaka	Značenje
b_{in}/b_{out}	Broj ulaznih/izlaznih bitova VHDL programa
N_{test}	Broj provedenih testova
$N_{\leq rqc}$	Broj testova u kojima je kvantni trošak generiranog sklopovlja mani ili jednak kvantnom trošku reverzibilnog sklopovlja
$N_{>rqc}$	Broj testova u kojima je kvantni trošak generiranog sklopovlja veći od kvantnoga troška reverzibilnog sklopovlja
rcl_{min}	Minimalna duljina reverzibilnog sklopovlja
rcl_{max}	Maksimalna duljina reverzibilnog sklopovlja
rcl_{avg}	Prosječna duljina reverzibilnog sklopovlja
$qclen_{min}$	Minimalna duljina kvantnog sklopovlja
$qclen_{max}$	Maksimalna duljina kvantnog sklopovlja
$qclen_{avg}$	Prosječna duljina kvantnog sklopovlja
Nbr_{min}	Minimalni broj bitova reverzibilnog sklopovlja
Nbr_{max}	Maksimalni broj bitova reverzibilnog sklopovlja
Nbr_{avg}	Prosječni broj bitova reverzibilnog sklopovlja

t_{min} (µs)	Minimalno vrijeme pojedinog dijela testa u mikrosekundama
t_{max} (µs)	Maksimalno vrijeme pojedinog dijela testa u mikrosekundama
t_{avg} (µs)	Prosječno vrijeme pojedinog dijela testa u mikrosekundama
qc_{min}	Minimalni kvantni trošak
qc_{max}	Maksimalni kvantni trošak
qc_{avg}	Prosječni kvantni trošak
acc_{min} (%)	Minimalna točnost pretvorbe VHDL izvornoga koda u QCL izvorni kod u postocima
acc_{max} (%)	Maksimalna točnost pretvorbe VHDL izvornoga koda u QCL izvorni kod u postocima
acc_{avg} (%)	Prosječna točnost pretvorbe VHDL izvornoga koda u QCL izvorni kod u postocima

8.3.1. Prevođenje VHDL koda u reverzibilno sklopovlje

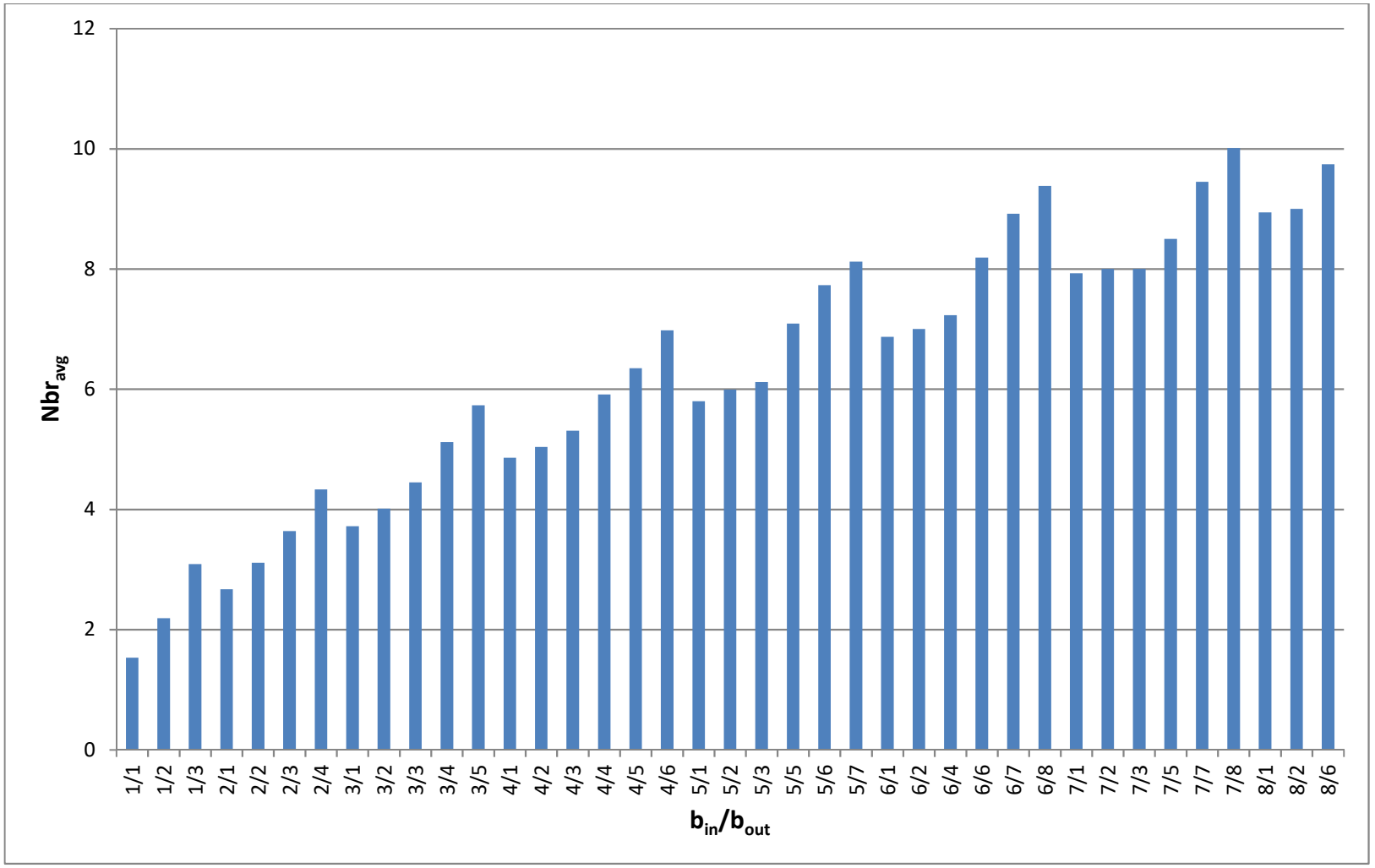
Ono što je zajedničko svim grupama testova je prvi korak u procesu prevođenja (slika 39), bez obzira koji algoritam se koristio za generiranje kvantnog sklopovlja. Prvi korak u procesu prevođenja je pretvaranje VHDL koda u reverzibilno sklopovlje upotrebom proširenih tablica istinitosti. Rezultati i analiza tog koraka je u nastavku.

Tablica 19: Rezultati testiranja prevođenja izvornoga VHDL koda u reverzibilno sklopovlje

b_{in}/b_{out}	N_{test}	Nbr_{min}	Nbr_{max}	Nbr_{avg}	rcl_{min}	rcl_{max}	rcl_{avg}	t_{min} (µs)	t_{max} (µs)	t_{avg}(µs)
1/1	100	1	2	1,53	0	1	0,49	260347	333363	276464,17
1/2	100	2	3	2,19	513	748	609,05	584470	709735	600278,81
1/3	100	3	4	3,09	50	791	431	600078	657522	615013,45
2/1	100	2	3	2,67	0	4	2,09	270698	383879	290140,3
2/2	100	2	4	3,11	0	11	2,62	282622	500395	302935,41
2/3	100	3	5	3,64	54	619	248,99	619332	795381	643291,83
2/4	100	4	6	4,33	385	766	559,26	633314	732904	655104,04
3/1	100	3	4	3,72	1	10	5,37	297176	407954	311138,51
3/2	100	3	5	4,01	0	10	5,44	312892	398830	330797,37
3/3	100	3	6	4,45	1	14	6,8	330763	444167	349471,19
3/4	100	4	6	5,12	445	794	661,67	673916	773660	693199,07
3/5	100	5	7	5,73	477	1729	1246,61	694959	785782	722364,09
4/1	100	4	5	4,86	2	21	8,87	344962	457670	380540,49
4/2	100	5	6	5,04	81	329	241,23	480016	1897149	532042,41
4/3	100	5	6	5,31	4	25	14,04	448110	633525	482825,61
4/4	100	5	7	5,91	4	33	16,55	474997	704472	517099,66
4/5	100	5	7	6,35	108	1838	930,82	771640	901174	814216,04
4/6	100	6	8	6,98	118	1841	826,08	807404	1122454	899142,37
5/1	100	5	6	5,8	114	319	215,31	510410	572736	526363,58
5/2	100	5	6	5,99	3	60	24,14	537493	638060	565773,62
5/3	100	6	7	6,12	4	53	18,22	590257	896447	660006,05

5/5	100	6	8	7,09	15	61	37,82	648835	1184779	826013,77
5/6	100	7	9	7,73	1004	1789	1311,35	990112	2405122	1336334,26
5/7	100	7	9	8,12	1154	1833	1561,32	1144613	2637959	1590643,66
6/1	100	6	7	6,87	23	333	159,96	619561	686690	639059,56
6/2	100	7	7	7	9	58	31,66	773874	1250552	927014,92
6/4	100	7	8	7,23	23	68	44,48	1040133	1897560	1259525,19
6/6	100	8	9	8,19	34	136	88,42	1684156	3931400	2249795,75
6/7	100	8	10	8,92	1190	4027	2759,82	2089754	9172400	3853078,78
6/8	100	9	10	9,38	2359	4059	3638,31	3403182	9541677	5946498,16
7/1	100	7	8	7,93	24	330	162,8	880963	1090423	921399,46
7/2	100	8	8	8	169	688	313,91	1992787	2900243	2520580,48
7/3	100	8	8	8	35	136	82,68	2574845	3271136	2887155,41
7/5	100	8	9	8,5	42	139	97,21	3110652	7299595	4925248,27
7/7	100	9	10	9,45	8	146	29,06	5852425	17340991	10939862,93
7/8	100	9	11	10,01	2924	8869	3852,61	7588606	42059743	17045561,36
8/1	100	8	9	8,94	31	147	99,82	1972800	3331197	2132202,01
8/2	100	9	9	9	453	804	658,5	8322263	12004524	10182573,67
8/6	100	9	10	9,74	82	324	147,55	13047165	33182713	26313759,56

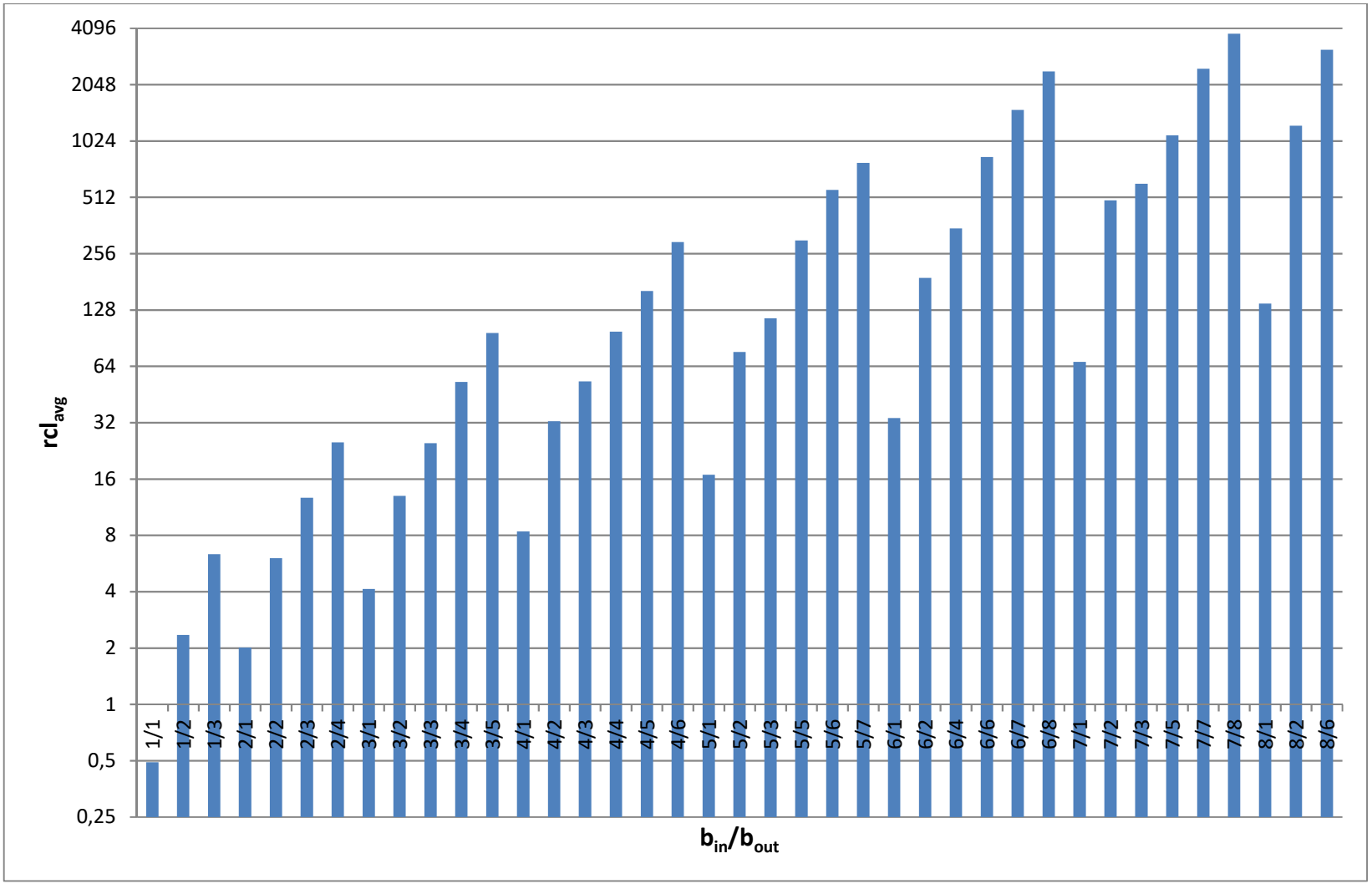
Sljedeći grafikon prikazuje prosječni broj bitova reverzibilnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa na linearnoj skali.



Grafikon 8: Prosječni broj bitova reverzibilnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa

Na prethodnom grafikonu primjetan je linearan rast prosječnog broja bitova u reverzibilnom sklopovlju u situacijama kada se povećava broj ulaznih ili izlaznih bitova. Taj rast je izraženiji u situacijama kada je broj ulaznih bitova fiksan (npr. b_{in} = 4), a broj izlaznih bitova je veći od broja ulaznih bitova (npr. b_{out} = 5, 6). Taj rezultat je očekivan zbog toga što u tim slučajevima raste broj $max(b_{in}, b_{out})$ koji određuje broj bitova u reverzibilnom sklopovlju zbog pravila da broj ulaznih i izlaznih bitova u reverzibilnom sklopovlju mora biti jednak. Ista situacija je i u obrnutom slučaju kada je fiksan broj izlaznih bitova, a raste broj ulaznih bitova, pri čemu je broj ulaznih bitova veći od fiksnog broja izlaznih bitova (npr. b_{in} = 2, 3, 4, 5, 6, 7, 8, b_{out} = 1). U situacijama u kojima je broj ulaznih bitova u VHDL program fiksan (npr. b_{in} = 5), a broj izlaznih bitova raste (b_{out} = 1, 2, 3), pri čemu je broj ulaznih bitova veći od broja izlaznih bitova, primjetan je sporiji rast u odnosu na slučaj u kojem je broj izlaznih bitova veći od broja ulaznih bitova. Važno je napomenuti da će broj bitova reverzibilnog sklopovlja u daljnjem postupku pretvorbe odrediti broj kvantnih bitova kvantnog sklopovlja, te dimenziju unitarne matrice koja će se dekomponirati QR i CS algoritmima.

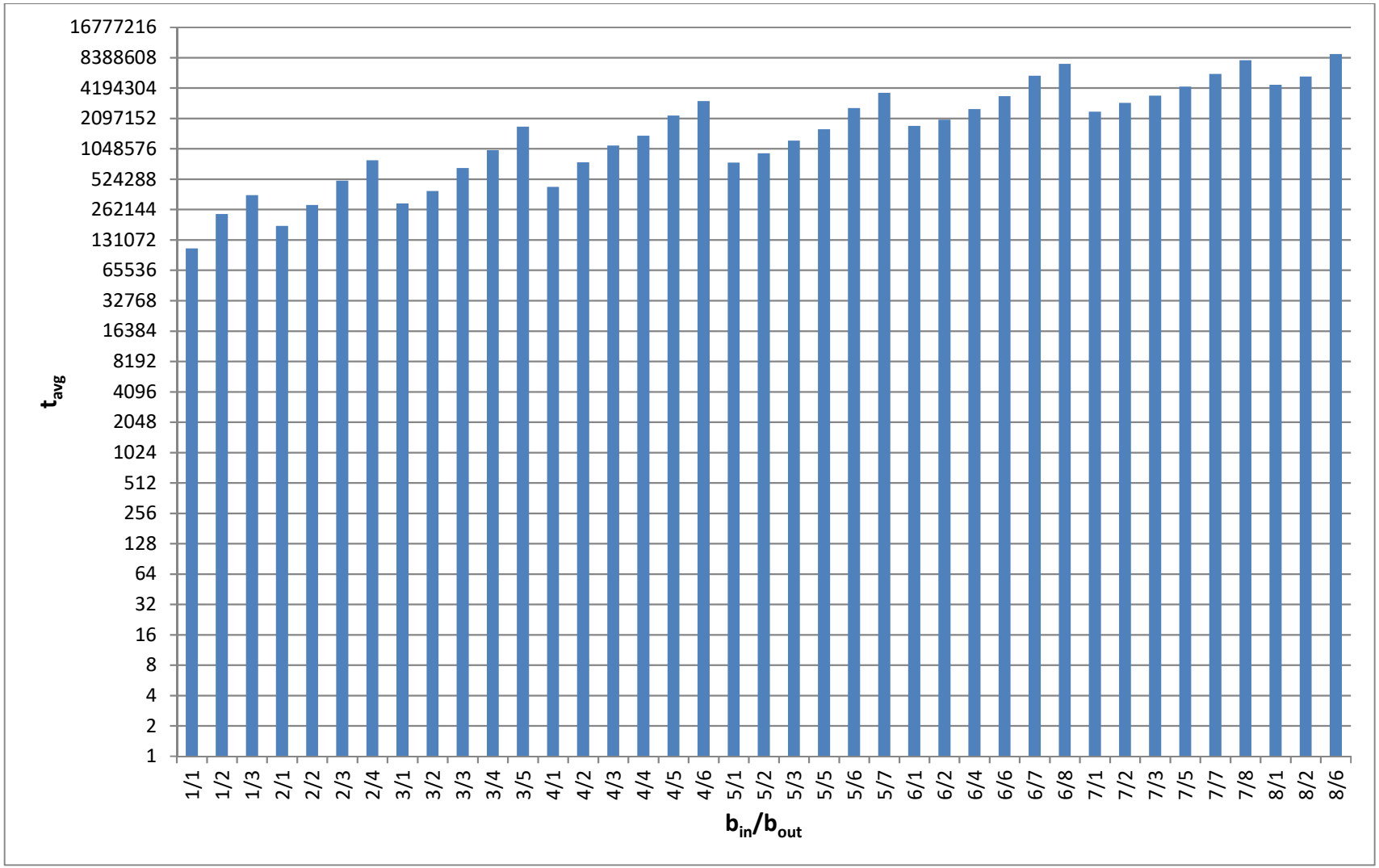
Sljedeći grafikon prikazuje prosječnu duljinu reverzibilnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 9: Prosječna duljina reverzibilnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa

Na prethodnom grafikonu primjetna je eksponencijalna ovisnost duljine reverzibilnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova iz VHDL programa, što je razumljivo zbog načina funkcioniranja Millerovog, Maslovog i Dueckovog algoritma koji se temelji na redovima u proširenoj tablici istinitosti, čiji broj pak eksponencijalno raste s povećanjem broja ulaznih bitova, odnosno eksponencijalno raste i broj potrebnih kvantnih vrata prilikom obrade svakog retka reverzibilne tablice istinitosti prilikom povećanja broja izlaznih bitova iz VHDL programa. Konkretno, fiksiramo li broj ulaznih bitova (npr. b_{in} = 3), a broj izlaznih bitova (npr. b_{out} = 1, 2, 3, 4, 5) raste, primjetan je eksponencijalni rast prosječne duljine reverzibilnog sklopovlja na prethodnom grafu. Zanimljivo je uočiti da je prosječna duljina reverzibilnog sklopovlja u testovima (npr. b_{in} = 3, b_{out} = 4) vrlo slična testovima u kojima se broj ulaznih i izlaznih bitova zamijeni (npr. b_{in} = 4, b_{out} = 3).

Sljedeći grafikon prikazuje prosječno vrijeme generiranja reverzibilnog sklopovlja iz VHDL koda u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 10: Prosječno vrijeme trajanja generiranja reverzibilnog sklopovlja iz VHDL koda u odnosu na broj ulaznih i izlaznih bitova VHDL programa

Na prethodnom grafu očekivano je primjetna eksponencijalna ovisnost prosječnog vremena za generiranje reverzibilnog sklopovlja na temelju VHDL koda o broju ulaznih odnosno izlaznih bitova. Dakle, potrebno vrijeme za generiranje tablice istinitosti na temelju VHDL koda eksponencijalno ovisi o broju ulaznih bitova u VHDL program zato što se na temelju svakog mogućeg ulaza u VHDL program računa pripadajući izlaz. Potrebno vrijeme za proširivanje tablice istinitosti i generiranje reverzibilnog sklopovlja na temelju proširene tablice istinitosti također je eksponencijalno ovisno o broju ulaznih odnosno izlaznih bitova zbog toga što Millerov, Maslovov i Dueckov algoritam prolazi kroz svaki redak proširene tablice istinitosti i generira potrebna kvantna vrata. Broj redaka u proširenoj tablici istinitosti eksponencijalno ovisi o broju ulaznih bitova, dok broj generiranih vrata, koja će Millerov, Maslovov i Dueckov algoritam generirati za pojedini redak, eksponencijalno ovisi o broju izlaznih bitova.

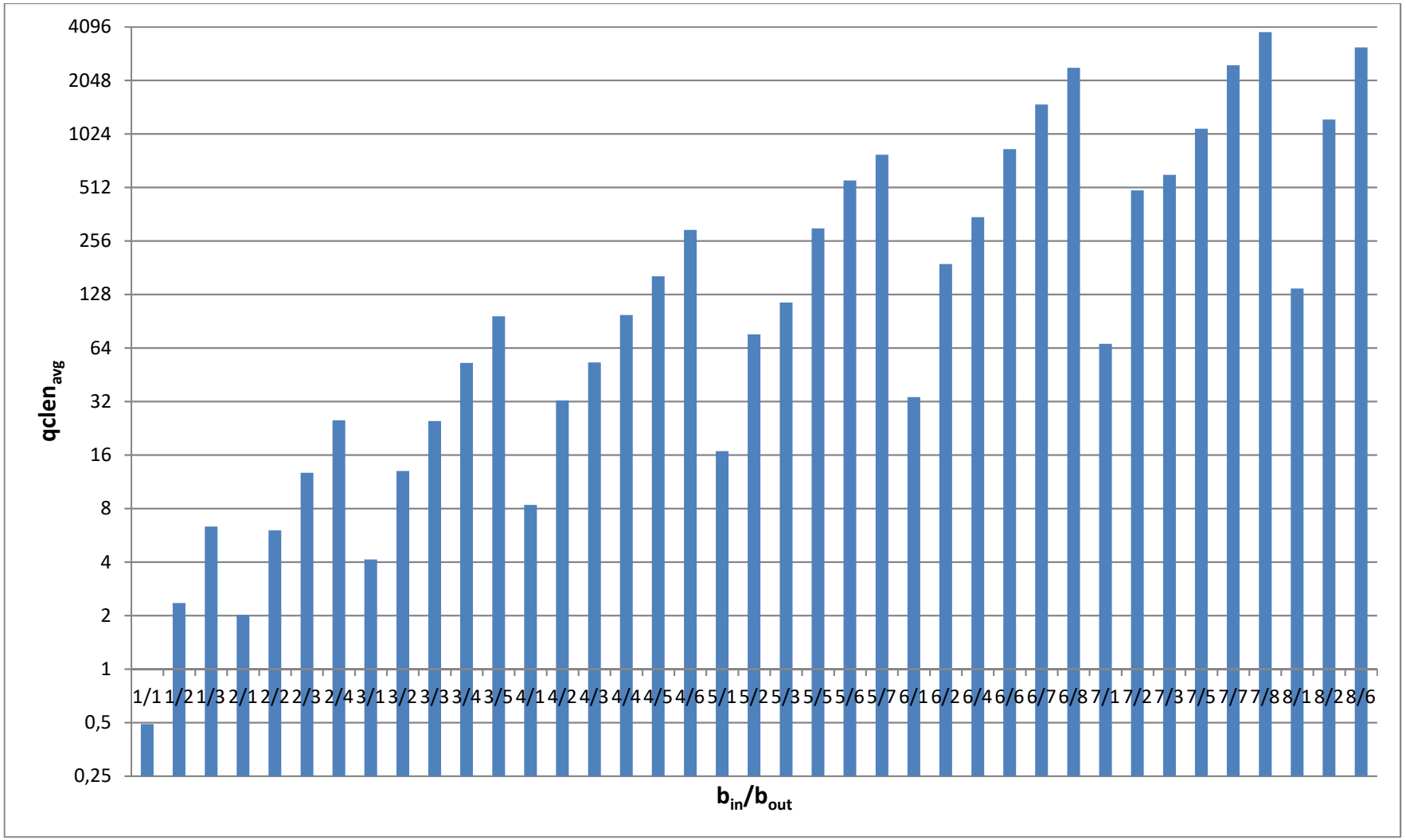
8.3.2. Algoritam za izravno pretvaranje reverzibilnog sklopovlja u kvantno sklopovlje

Tablica 20: Rezultati testiranja prevođenja reverzibilnog sklopovlja u kvantno sklopovlje, te kvantnog sklopovlja u QCL kod upotrebom algoritma za izravno pretvaranje reverzibilnog sklopovlja u kvantno sklopovlje

b_{in} /b_{out}	N_{test}	$N_{\leq rqc}$	$N_{>rqc}$	$qclen_{min}$	$qclen_{max}$	$qclen_{avg}$	t_{min} (µs)	t_{max} (µs)	t_{avg}(µs)	qc_{min}	qc_{max}	qc_{avg}	acc_{min} (%)	acc_{max} (%)	acc_{avg} (%)
1/1	100	100	0	0	1	0,49	96	256	133,28	0	1	0,49	100	100	100
1/2	100	100	0	513	748	609,05	84	319	183,26	0	4	2,35	100	100	100
1/3	100	100	0	50	791	431	137	509	271,01	0	30	17,02	100	100	100
2/1	100	100	0	0	4	2,09	94	24749	752,79	0	8	4,29	100	100	100
2/2	100	100	0	0	11	2,62	121	24672	849,58	1	88	22,39	100	100	100
2/3	100	100	0	54	619	248,99	197	1176	397,27	3	676	93,11	100	100	100
2/4	100	100	0	385	766	559,26	277	1657	639,61	24	3855	333,65	100	100	100
3/1	100	100	0	1	10	5,37	147	30436	604,18	2	35	19,69	100	100	100
3/2	100	100	0	0	10	5,44	169	23492	1498,82	7	527	110,8	100	100	100
3/3	100	100	0	1	14	6,8	222	23509	1082,24	21	1859	373,69	100	100	100
3/4	100	100	0	445	794	661,67	381	2751	1260,19	68	6093	1475,15	100	100	100
3/5	100	100	0	477	1729	1246,61	666	5694	2209,46	458	29644	4642,56	100	100	100
4/1	100	100	0	2	21	8,87	191	60566	2113,07	13	143	86,01	100	100	100
4/2	100	100	0	81	329	241,23	249	1144	509,59	111	2227	637,8	100	100	100
4/3	100	100	0	4	25	14,04	426	35083	1698,75	311	6016	1623,51	100	100	100
4/4	100	100	0	4	33	16,55	595	25833	1822,76	655	28098	4867,45	100	100	100
4/5	100	100	0	108	1838	930,82	1304	5404	2985,38	1195	35440	12815,37	100	100	100
4/6	100	100	0	118	1841	826,08	1529	13307	5014,3	4306	184189	41015,06	100	100	100
5/1	100	100	0	114	319	215,31	225	1180	332,7	91	609	350,74	100	100	100
5/2	100	100	0	3	60	24,14	675	38403	2145,45	785	5479	3498,24	100	100	100
5/3	100	100	0	4	53	18,22	1149	33752	2800,11	2794	25425	6781,02	100	100	100
5/5	100	100	0	15	61	37,82	1605	11584	5060,27	4502	183468	41475,87	100	100	100
5/6	100	100	0	1004	1789	1311,35	3499	32314	10199,68	20714	899836	138466,47	100	100	100

5/7	100	100	0	1154	1833	1561,32	6181	39042	15100,81	31877	992133	265839,57	100	100	100
6/1	100	100	0	23	333	159,96	359	1394	538,09	660	3089	1514,08	100	100	100
6/2	100	100	0	9	58	31,66	1562	20980	3418,31	6845	27824	21323,65	100	100	100
6/4	100	100	0	23	68	44,48	3394	49647	6822,53	22987	171750	55809,33	100	100	100
6/6	100	100	0	34	136	88,42	9638	97948	17665,69	137913	983535	299748,88	100	100	100
6/7	100	100	0	1190	4027	2759,82	12121	109415	32282,33	152019	5104088	867081,18	100	100	100
6/8	100	100	0	2359	4059	3638,31	29005	109951	60552,87	758191	5182655	2374256,08	100	100	100
7/1	100	100	0	24	330	162,8	686	2280	1006,34	3154	16144	6107,81	100	100	100
7/2	100	100	0	169	688	313,91	5887	11232	8765,21	96092	167691	132838,78	100	100	100
7/3	100	100	0	35	136	82,68	9658	67720	12250,17	115840	183462	150945,07	100	100	100
7/5	100	100	0	42	139	97,21	12161	37079	22681,05	150057	1039000	515226,91	100	100	100
7/7	100	100	0	8	146	29,06	28843	110618	64005,25	713699	5254913	2630830,58	100	100	100
7/8	100	100	0	2924	8869	3852,61	43400	646239	114916,9	885260	26352413	5266604,18	100	100	100
8/1	100	100	0	31	147	99,82	2463	7898	3355,7	12274	101840	25984,76	100	100	100
8/2	100	100	0	453	804	658,5	20094	32939	26121,5	639540	925108	775403,97	100	100	100
8/6	100	100	0	82	324	147,55	39364	126889	85519,7	828556	5370703	3796800,65	100	100	100

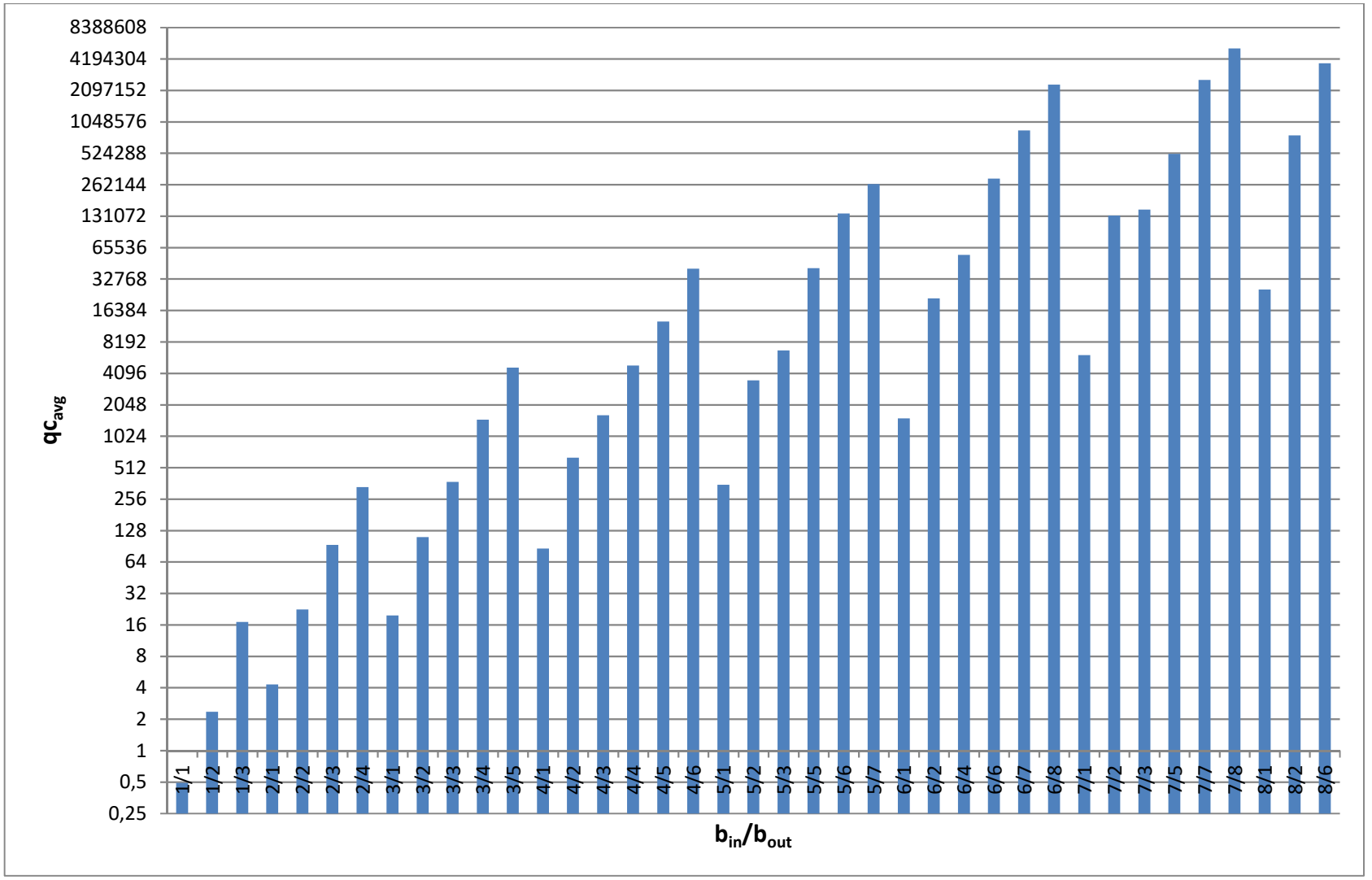
U ovome dijelu rada analizirat ću korake pretvorbe reverzibilnog sklopovlja u kvantno sklopovlje upotrebom algoritma za izravnu pretvorbu, te generiranje QCL koda na temelju generiranog kvantnog sklopovlja. Sljedeći grafikon prikazuje prosječnu duljinu generiranog kvantnog sklopovlja prilikom izravne pretvorbe reverzibilnog sklopovlja u kvantno sklopovlje u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 11: Prosječna duljina kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (algoritam za izravno pretvaranje reverzibilnog sklopovlja u kvantno)

S obzirom da se prilikom generiranja kvantnog sklopovlja upotrebom algoritma za izravnu pretvorbu sva reverzibilna vrata reverzibilnog sklopovlja pretvaraju u kvantna vrata, duljina kvantnog sklopovlja je u biti jednaka duljini reverzibilnog sklopovlja, što je i primjetno na grafikonu 11 koji je identičan grafikonu 9. Kako duljina reverzibilnog sklopovlja eksponencijalno ovisi o broju ulaznih i izlaznih bitova VHDL programa, možemo zaključiti da i duljina kvantnog sklopovlja koje je generirano izravnom pretvorbom iz reverzibilnog sklopovlja eksponencijalno ovisi o broju ulaznih i izlaznih bitova VHDL programa, što je i vidljivo iz prethodnog grafikona.

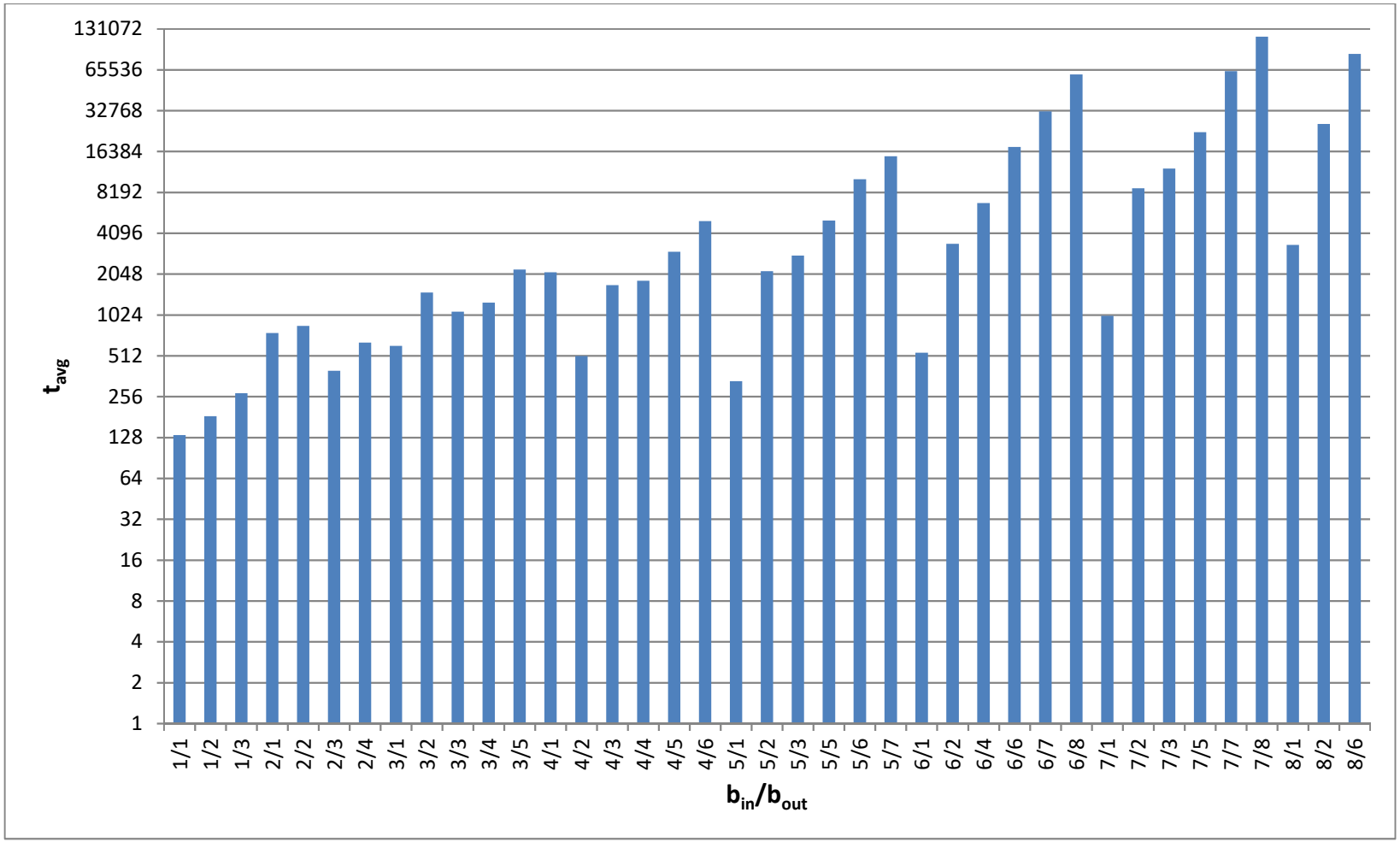
Sljedeći grafikon prikazuje prosječni kvantni trošak generiranog kvantnog sklopovlja prilikom izravne pretvorbe reverzibilnog sklopovlja u kvantno sklopovlje u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 12: Prosječni kvantni trošak kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (algoritam za izravno pretvaranje reverzibilnog sklopovlja u kvantno sklopovlje)

Duljina kvantnog sklopovlja koje je generirano ovim algoritmom jednaka je duljini reverzibilnog sklopovlja na temelju kojeg je kvantno sklopovlje i generirano, te sama duljina kvantnog sklopovlja utječe na ukupni kvantni trošak generiranog kvantnog sklopovlja. Ukoliko pogledamo grafikon 11, koji opisuje duljinu kvantnog sklopovlja, lako možemo uočiti da je trend tog grafikona sličan trendu grafikona 12, što znači da je kvantni trošak ovog algoritma ustvari polinomno ovisan o samoj duljini kvantnog sklopovlja. S obzirom da je duljina kvantnog sklopovlja koje je generirano ovim algoritmom eksponencijalno ovisna o broju ulaznih odnosno izlaznih bitova VHDL programa, možemo zaključiti da je kvantni trošak kvantnih sklopovlja koji su generirani ovim algoritmom ovisan o broju ulaznih i izlaznih bitova, što je i vidljivo na prethodnom grafikonu. Dakako, kod ovog algoritma je kvantni trošak generiranih kvantnih sklopova jednak kvantnom trošku reverzibilnih sklopova zbog izravnog pretvaranja reverzibilnih vrata u kvantna vrata.

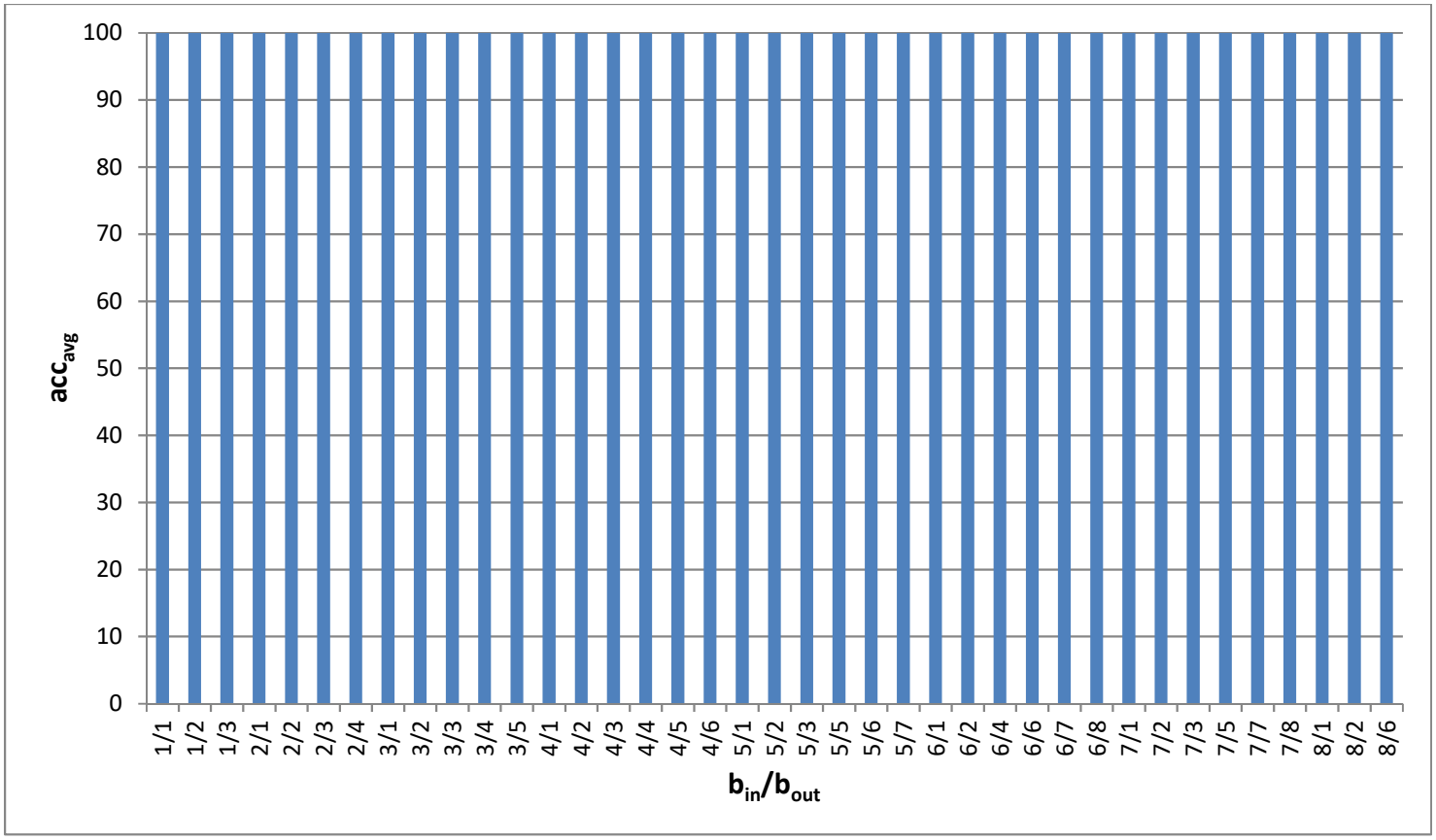
Sljedeći grafikon prikazuje prosječno vrijeme izvođenja preostalih koraka implementiranog modela (generiranje kvantnog sklopovlja na temelju reverzibilnog sklopovlja, te generiranje QCL koda na temelju kvantnog sklopovlja) upotrebom algoritma za izravno prevođenje reverzibilnog sklopovlja u kvantno sklopovlje u odnosu na broj ulaznih i izlaznih bitova VHDL programa. Vrijednosti su prikazane na logaritamskoj skali.



Grafikon 13: Prosječno vrijeme trajanja generiranja kvantnog sklopovlja na temelju reverzibilnog sklopovlja, te generiranja QCL koda na temelju kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (algoritam za izravno pretvaranje reverzibilnog sklopovlja u kvantno)

Na prethodnom grafikonu primjetna je eksponencijalna ovisnost prosječnog vremena trajanja generiranja kvantnog sklopovlja upotrebom algoritma za izravno prevođenje, te prevođenja kvantnog sklopovlja u QCL kod u odnosu na broj ulaznih i izlaznih bitova VHDL programa. Vremensku složenost možemo bolje shvatiti ako zasebno sagledamo preostale korake implementiranog modela sa slike 39 (generiranje kvantnog sklopovlja na temelju reverzibilnog sklopovlja, te generiranje QCL koda na temelju kvantnog sklopovlja). Složenost koraka u kojem se reverzibilno sklopovlje pretvara u kvantno sklopovlje polinomno ovisi o duljini reverzibilnog sklopovlja, zato što algoritam prolazi kroz sva reverzibilna vrata i izravno ih pretvara u kvantna vrata. Međutim, duljina reverzibilnog sklopovlja, kao što smo već napomenuli, eksponencijalno ovisi o broju bitova reverzibilnog sklopovlja koji pak ovisi o broju ulaznih i izlaznih bitova VHDL programa (grafikon 8). Složenost koraka u kojem se kvantno sklopovlje pretvara u naredbe QCL jezika polinomno ovisi o duljini kvantnog sklopovlja. Dakako, ukupna složenost opisana dva koraka je eksponencijalna u odnosnu na broj ulaznih i izlaznih bitova VHDL programa, što je i vidljivo na prethodnom grafikonu.

Sljedeći grafikon prikazuje prosječnu točnost generiranog QCL koda upotrebom algoritma za izravno pretvaranje reverzibilnog sklopovlja u kvantno u odnosu na broj ulaznih i izlaznih bitova VHDL programa.



Grafikon 14: Prosječna točnost generiranog QCL koda u odnosu na broj ulaznih i izlaznih bitova VHDL programa (algoritam za izravno pretvaranje reverzibilnog sklopovlja u kvantno)

Sve instance testova su uspješno prošle s gledišta točnosti generiranog QCL koda. Možemo, dakle, zaključiti da prilikom pretvorbe VHDL koda u QCL kod nije došlo do pogreške zbog nužnog zaokruživanja koje je bilo postavljeno na 10 decimala. Važno je napomenuti da se upotrebom algoritma za izravno pretvaranje reverzibilnog sklopovlja osigurava upotreba cjelobrojne aritmetike u cijelom procesu prevođenja što

u konačnici znači da će točnost generiranog QCL koda uvijek biti maksimalna, bez obzira na broj ulaznih i izlaznih bitova izvornoga VHDL programa te postavljenog zaokruživanja.

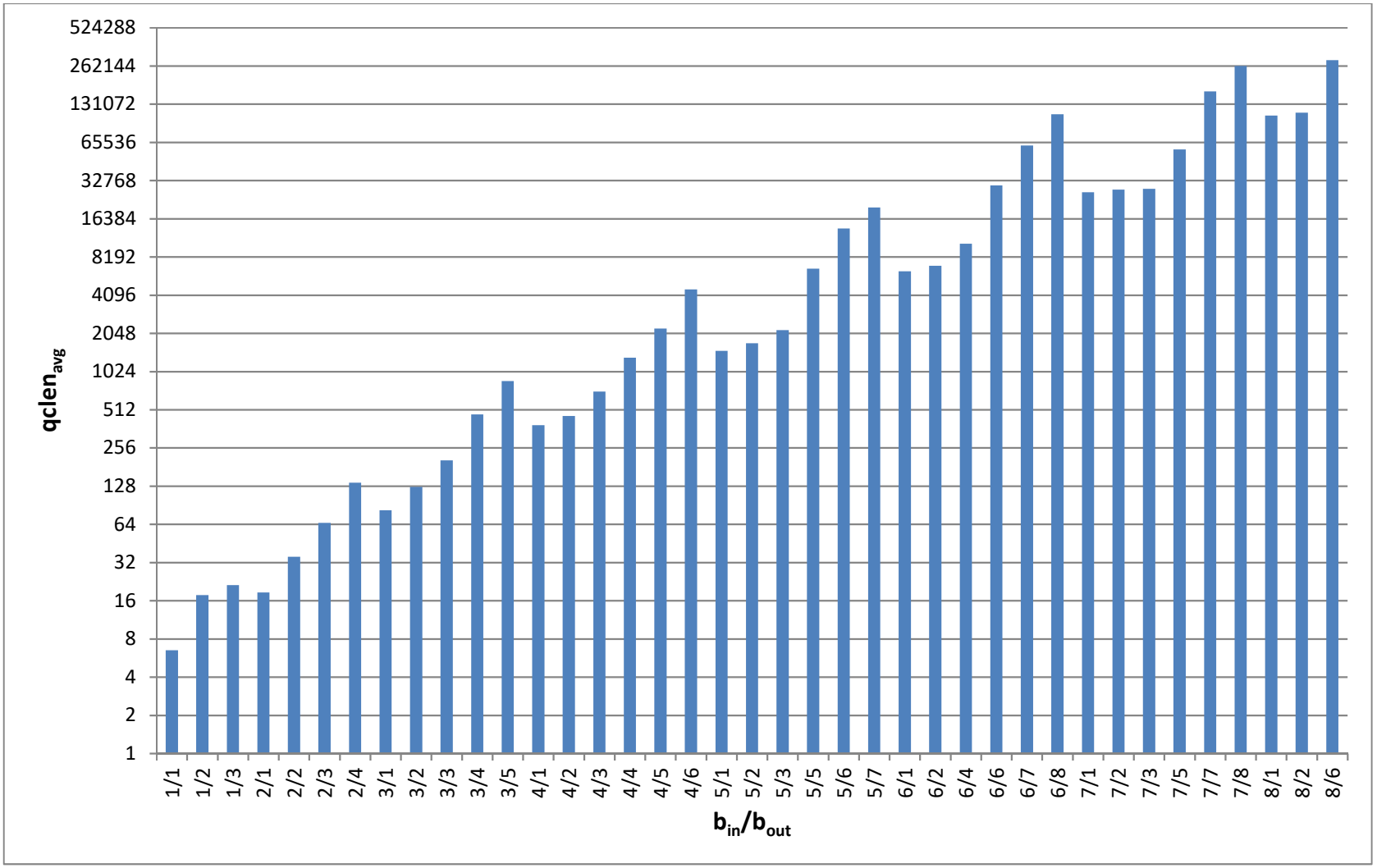
8.3.3. QR dekompozicija

Tablica 21: Rezultati testiranja prevođenja izvornoga VHDL koda u QCL kod upotrebom QR dekompozicije

b_{in} /b_{out}	N_{test}	$N_{\leq rqc}$	$N_{>rqc}$	$qclen_{min}$	$qclen_{max}$	$qclen_{avg}$	t_{min} (µs)	t_{max} (µs)	t_{avg} (µs)	qc_{min}	qc_{max}	qc_{avg}	acc_{min} (%)	acc_{max} (%)	acc_{avg} (%)
1/1	100	51	49	0	26	6,5	139	737	299,32	0	8	2,48	100	100	100
1/2	100	43	57	0	208	17,7	185	5076	796,01	0	139	19,64	100	100	100
1/3	100	9	91	0	75	21,26	308,5	6483	1524,775	0	838	117,59	100	100	100
2/1	100	12	88	0	48	18,58	147	1426	614,25	0	172	60,47	100	100	100
2/2	100	3	97	3	134	35,54	251	4336	1220,12	3	1507	270,52	100	100	100
2/3	100	2	98	3	313	65,77	408	25031	3752,85	12	10322	1049,92	100	100	100
2/4	100	0	100	17	711	135,86	1698	149543	10395,14	208	69028	3804,86	100	100	100
3/1	100	0	100	3	204	82,34	397	5785	2629,64	22	2286	869,48	100	100	100
3/2	100	0	100	17	571	126,37	658	30708	5580,09	58	18806	2435,1	100	100	100
3/3	100	0	100	27	811	204,54	989	125487	13082,07	100	78957	6476,81	100	100	100
3/4	100	0	100	71	1343	470,2	3371	257411	64712,93	806	130404	30243,64	100	100	100
3/5	100	0	100	201	3218	861,65	17009	2172840	234416,15	6620	927409	105005,39	100	100	100
4/1	100	0	100	82	736	385,73	3343	49611	23186,59	912	24281	12330,59	100	100	100
4/2	100	0	100	226	1873	457,66	14837	261468	32375,88	7433	181752	18497,33	100	100	100
4/3	100	0	100	282	1958	711,59	21103	329382	97354,03	9306	190120	50398,64	100	100	100
4/4	100	0	100	303	4457	1316,36	24794	2667502	312288,32	10002	1284950	155733,34	100	100	100
4/5	100	0	100	394	4903	2235,22	38042	3049268	1045326,46	12978	1413146	480706,38	100	100	100
4/6	100	0	100	989	13127	4536,05	187232	28786961	4930861,49	96164	11296528	2079948,93	100	100	100
5/1	100	0	100	286	2535	1490,39	17541	344103	205943,5	9364	245947	138772,52	100	100	100

5/2	100	0	100	496	2241	1707,21	37675	372000	295963,48	16376	217576	165382,3	100	100	100
5/3	100	0	100	1129	6909	2165,62	217361	3827030	657586,1	109584	1991260	336187,63	100	100	100
5/5	100	0	100	1448	18180	6610,07	276441	39896309	7075155,87	140636	15645030	2977713,66	100	100	100
5/6	100	0	100	3740	49541	13723,05	2035990	357579098	29574662,07	1078446	127639034	12490198,93	100	100	100
5/7	100	0	100	5447	49837	20078,26	2779339	443995304	79000995,24	1569610	128401482	28105697,95	100	100	100
6/1	100	0	100	1265	9404	6315,64	224670	4758220	3311763,05	122722	2710438	1777326,69	100	100	100
6/2	100	0	100	5559	8682	6984,72	3183737	4799426	3927450,78	1601600	2502497	2013098,75	100	100	100
6/4	100	0	100	5188	26486	10395,27	2623811	51703543	13094259,83	1495036	22792446	5940671,02	100	100	100
6/6	100	0	100	18006	72308	29959,46	31819450	593113053	138354762,2	15496094	186297150	46690340,61	100	100	100
6/7	100	0	100	20434	167309	61913,11	37790465	1065614272	485358302	17585346	1292104399	164708268,2	100	100	100
6/8	100	0	100	57205	201678	108813,6	440874465	5256669641	1560690955	147389444	1556596714	634353255,7	100	100	100
7/1	100	0	100	6168	32980	26516,17	2818241	64345997	45385991,63	1776866	28380262	22541811,56	100	100	100
7/2	100	0	100	22255	32647	27746,95	39104407	64841065	51642855,12	19150615	28094524	23877882,82	100	100	100
7/3	100	0	100	23263	31994	28214,93	41136457	64895105	57049119,11	20018568	27532070	24280800,38	100	100	100
7/5	100	0	100	23059	100290	57665,68	42167522	861641093	377956498	19842830	258392656	124935880,1	100	100	100
7/7	100	0	100	75291	278083	165227	524184678	6939816060	2873955242	193985771	2147607330	1028359457	100	100	100
7/8	100	0	100	77136	431062	261822,8	621777638	9520495314	4364438357	198737628	2640077892	1992623032	100	100	100
8/1	100	0	100	23936	126919	106229,2	38852507	894406301	682370832,2	20595330	326990402	270866515,3	100	100	100
8/2	100	0	100	97186	126415	112056,5	353234814	963873632	701298323	250392308	325705190	288707045,9	100	100	100
8/6	100	0	100	104451	394366	291440,4	625990965	6967073613	2844346039	272783020	2912550306	1959042459	100	100	100

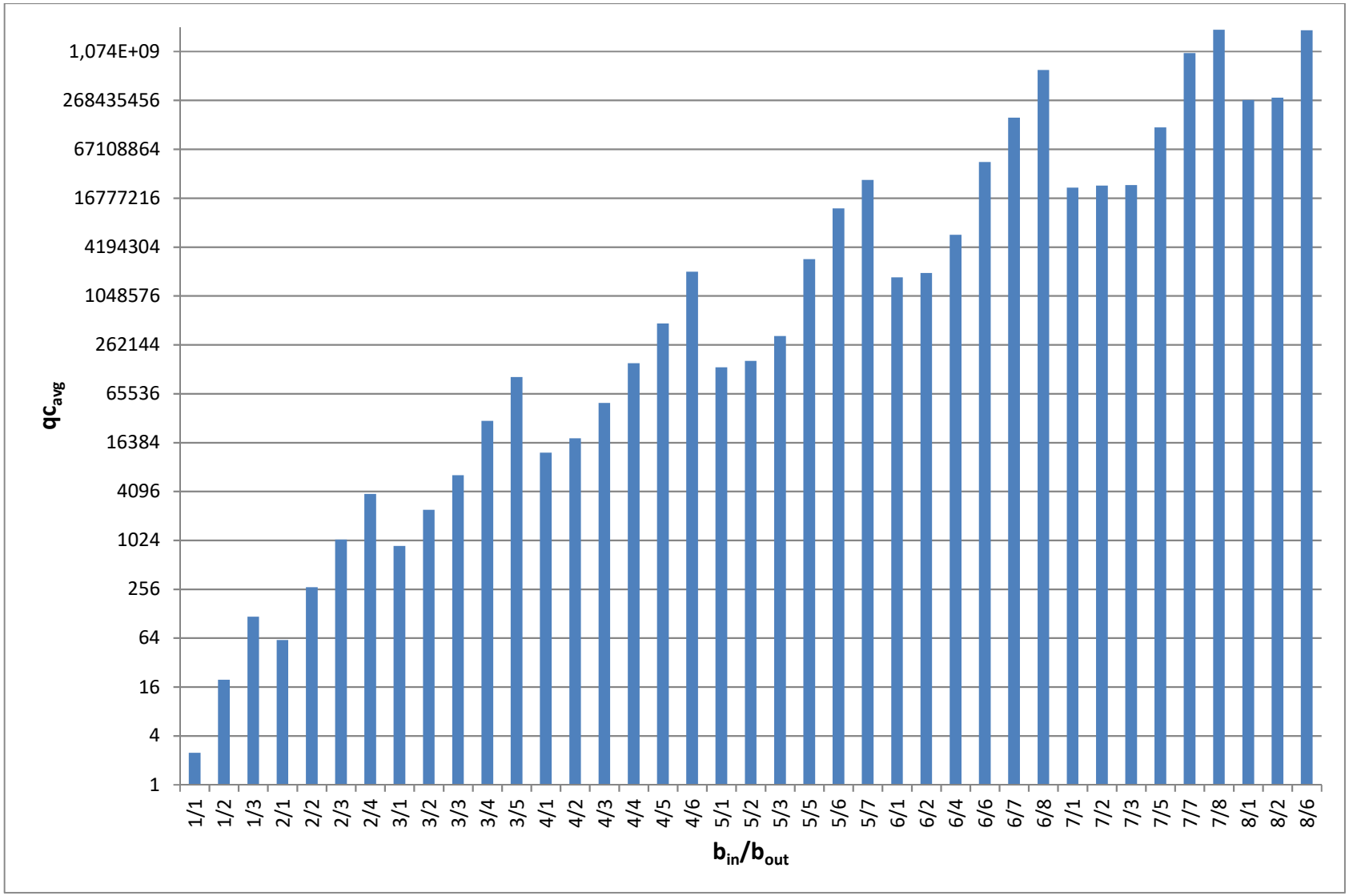
U ovom dijelu rada analizirat ću korake pretvorbe reverzibilnog sklopovlja u kvantno sklopovlje upotrebom QR algoritma, te generiranje QCL koda na temelju generiranog kvantnog sklopovlja. Sljedeći grafikon prikazuje prosječnu duljinu generiranog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 15: Prosječna duljina kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (QR algoritam)

Na prethodnom grafikonu primjetna je eksponencijalna ovisnost duljine generiranog kvantnog sklopovlja u ovisnosti o broju ulaznih i izlaznih bitova VHDL programa. Tu ovisnost možemo objasniti preko dimenzije unitarne matrice koja se dekomponira. Dimenzija unitarne matrice polinomno određuje broj elemenata u matrici, pa tako i broj elemenata ispod glavne dijagonale gdje se nalaze kandidati za nuliranje upotrebom Givensonovih rotacija. Za svaku Givensonovu rotaciju nastaju jedna kontrolna kvantna vrata viših dimenzija, koja se zatim dekomponiraju na kanonski oblik (slika 12), stoga duljina kvantnog sklopovlja koje je generirano QR algoritmom eksponencijalno ovisi o broju bitova reverzibilnog sklopovlja, odnosno o broju ulaznih i izlaznih bitova VHDL programa. Ta ovisnost je također uočljiva ako usporedimo grafikone 8 i 15 koji imaju sličan trend.

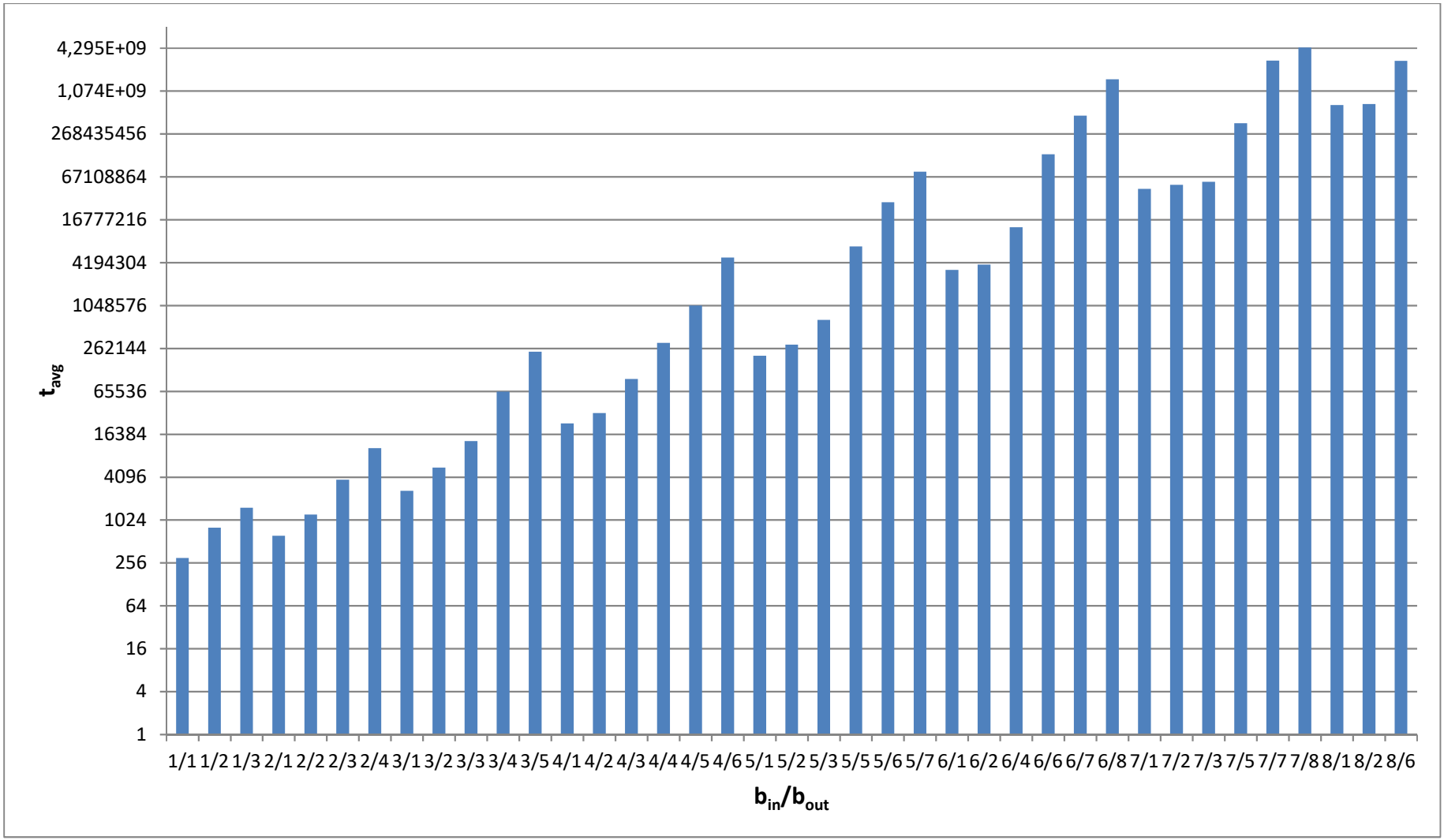
Sljedeći grafikon prikazuje prosječni kvantni trošak generiranog sklopovlja upotrebom QR algoritma u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 16: Prosječni kvantni trošak kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (QR algoritam)

S obzirom da se sve Givensonove rotacije pretvaraju u kvantna vrata viših dimenzija, koje se potom dekomponiraju na kanonski oblik (slika 12), za očekivati je da kvantni trošak ovisi o duljini generiranog kvantnog sklopovlja, što je uočljivo ako usporedimo trendove prethodnog grafikona s trendom grafikona 15. Možemo primijetiti da su trendovi ta dva grafikona slični te da kvantni trošak kvantnog sklopovlja koje je generirano QR algoritmom eksponencijalno ovisi o broju ulaznih i izlaznih bitova VHDL programa, što je i uočljivo na prethodnom grafikonu. Broj testova u kojima generirano sklopovlje ima manji ili jednak kvantni trošak reverzibilnom sklopovlju pada s porastom ulaznih odnosno izlaznih bitova. Iz kolona $\boldsymbol{N_{\leq rqc}}$ i $\boldsymbol{N_{>rqc}}$ u tablici 18 vidljivo je da ovaj algoritam može generirati kvantno sklopovlje koje ima manji ili jednak kvantni trošak kvantnom trošku reverzibilnog sklopovlja samo pri manjem broju ulaznih i izlaznih bitova.

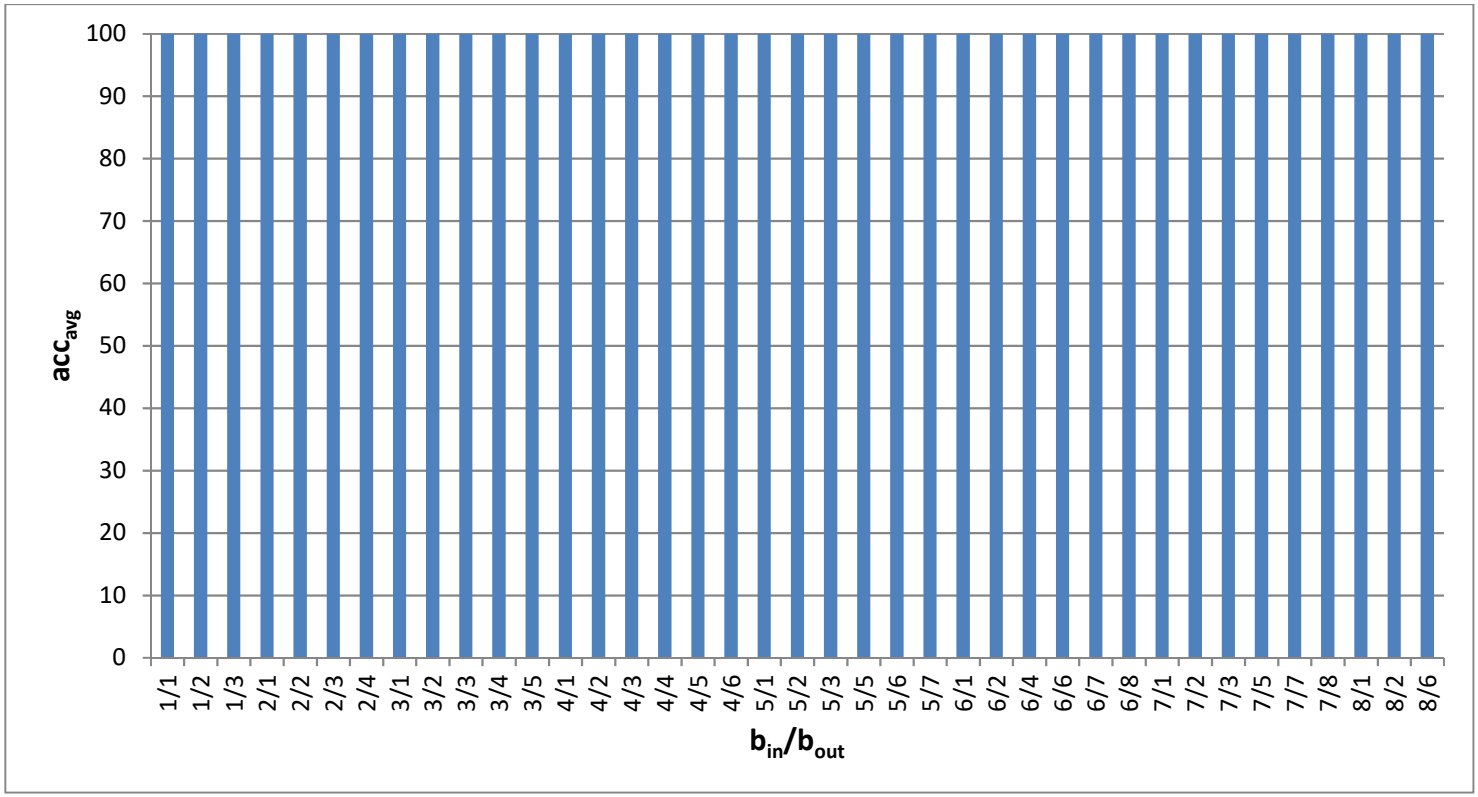
Sljedeći grafikon prikazuje prosječno vrijeme izvođenja preostalih koraka implementiranog modela (generiranje kvantnog sklopovlja na temelju reverzibilnog sklopovlja, te generiranje QCL koda na temelju kvantnog sklopovlja) upotrebom QR algoritma u odnosu na broj ulaznih i izlaznih bitova VHDL programa. Vrijednosti su prikazane na logaritamskoj skali.



Grafikon 17: Prosječno vrijeme trajanja preostalih koraka u odnosu na broj ulaznih i izlaznih bitova VHDL programa (QR dekompozicija)

Složenost koraka u kojem se reverzibilno sklopovlje pretvara u kvantno možemo podijeliti u dva dijela. Prvi dio je računanje unitarne matrice koja predstavlja logiku kvantnog sklopovlja, a drugi dio je dekompozicija te unitarne matrice i generiranje kvantnog sklopovlja. Dakle, množenje unitarnih matrica ima polinomnu složenost u ovisnosti o dimenziji unitarnih matrica, odnosno množenje unitarnih matrica ima eksponencijalnu ovisnost u odnosu na broj ulaznih i izlaznih bitova VHDL programa (grafikon 8). Osim toga, s obzirom da za potrebe računanja unitarne matrice reverzibilnog sklopovlja treba pomnožiti onoliko unitarnih matrica koliko ima reverzibilnih vrata u reverzibilnom sklopovlju, a duljina reverzibilnog sklopovlja je eksponencijalno ovisna o broju ulaznih i izlaznih bitova VHDL programa (grafikon 9), dolazimo do zaključka da je ukupna složenost računanja unitarne matrice eksponencijalno ovisna o broju ulaznih i izlaznih bitova. Drugi dio je dekompozicija unitarne matrice QR algoritmom. Složenost QR algoritma ovisi o dimenzijama matrice koja se dekomponira zbog potrebnih Givensonovih rotacija nad svakim elementom kojeg treba nulirati. Kako je broj elemenata koji se nuliraju Givensonovim rotacijama (elementi ispod glavne dijagonale) polinomno ovisan o dimenziji unitarne matrice koja se dekomponira, zaključujem da je složenost ovog koraka eksponencijalno ovisna o broju bitova u reverzibilnom sklopovlju, odnosno o broju ulaznih i izlaznih bitova VHDL programa koji direktno utječu na broj bitova reverzibilnog sklopovlja (grafikon 8). Dakako, prevođenje generiranog kvantnog sklopovlja u QCL kod polinomno ovisi o duljini kvantnog sklopovlja, koja je pak eksponencijalno ovisna o broju ulaznih i izlaznih bitova VHDL programa (grafikon 15). Ukupna vremenska složenost pretvorbe reverzibilnog sklopovlja u kvantno sklopovlje upotrebom QR algoritma te generiranje QCL koda na temelju generiranog kvantnog sklopovlja je eksponencijalno ovisna o broju ulaznih i izlaznih bitova, što je i uočljivo na prethodnom grafikonu.

Sljedeći grafikon prikazuje prosječnu točnost generiranog QCL izvornoga koda upotrebom QR algoritma u odnosu na broj ulaznih i izlaznih bitova VHDL programa.



Grafikon 18: Prosječna točnost generiranog QCL izvornoga koda u odnosu na broj ulaznih i izlaznih bitova VHDL programa (QR)

Kao i kod algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, tako su i kod upotrebe QR algoritma sve instance testova prošle uspješno s gledišta točnosti generiranog QCL koda, dakle možemo zaključiti da prilikom pretvorbe VHDL koda u QCL kod nije došlo do pogreške zbog nužnog zaokruživanja koje je bilo postavljeno na 10 decimalnih mjesta.

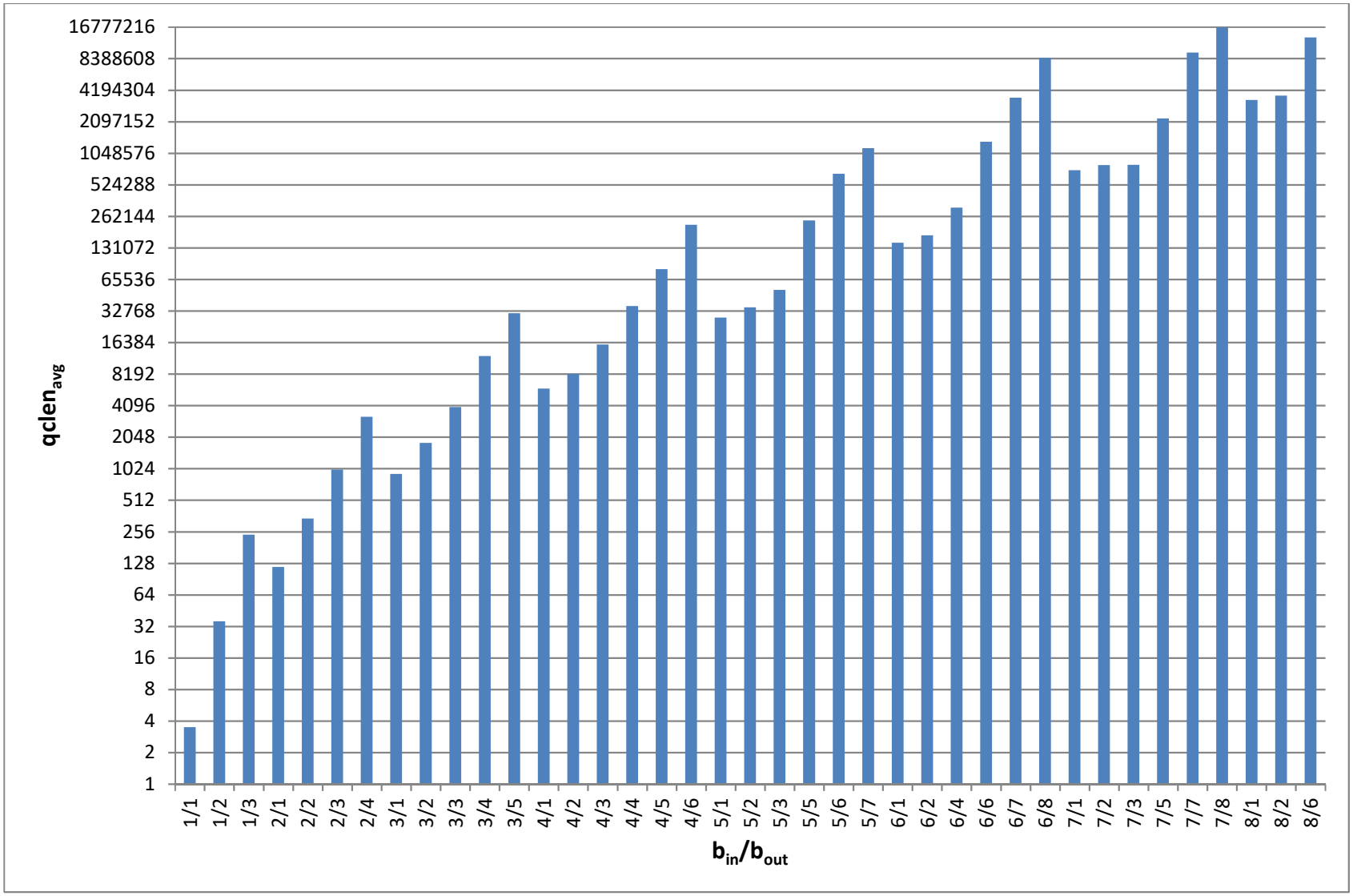
8.3.4. CS dekompozicija

Tablica 22: Rezultati testiranja prevođenja reverzibilnog sklopovlja u kvantno sklopovlje, te kvantnog sklopovlja u QCL kod upotrebom CS dekompozicije

b_{in} /b_{out}	N_{test}	$N_{\leq rqc}$	$N_{>rqc}$	$qclen_{min}$	$qclen_{max}$	$qclen_{avg}$	t_{min} (µs)	t_{max} (µs)	t_{avg} (µs)	qc_{min}	qc_{max}	qc_{avg}	acc_{min} (%)	acc_{max} (%)	acc_{avg} (%)
1/1	100	75	25	0	14	3,5	136	580	232,59	0	5	1,49	100	100	100
1/2	100	9	91	0	164	35,76	156	27076	1362,85	0	113	20,4	100	100	100
1/3	100	3	97	0	1172	239,66	169	26802	5796,03	0	1567	210,42	100	100	100
2/1	100	12	88	0	280	118	140	5883	2578,45	0	196	80,59	100	100	100
2/2	100	0	100	14	1506	342,86	486	31709	7332,49	5	2006	379,92	100	100	100
2/3	100	0	100	84	7934	1000,58	2215	177448	22633,08	57	22660	1778,52	100	100	100
2/4	100	0	100	376	25996	3202,04	8911	559175	68187,24	496	172918	8839,35	100	100	100
3/1	100	0	100	122	1738	910,52	2772	35361	18828,42	86	2318	1180,63	100	100	100
3/2	100	0	100	132	7756	1801,5	2963	159429	37632,98	93	22186	3672,97	100	100	100
3/3	100	0	100	262	25794	3969,64	5981	543290	83486,71	184	171588	11873,67	100	100	100
3/4	100	0	100	1074	37246	12191,96	22585	822470	260450,95	1434	247336	62922	100	100	100
3/5	100	0	100	5106	174582	31140,86	111591	3974586	688299,86	14763	2859446	277395,55	100	100	100
4/1	100	0	100	770	8414	5960,18	16337	172375	122228,35	1025	23991	16802,67	100	100	100
4/2	100	0	100	4384	34286	8178,5	91107	716922	169483,08	12598	228524	28426,87	100	100	100
4/3	100	0	100	5238	37188	15665,24	112974	793501	333169,03	14932	247983	85897,15	100	100	100
4/4	100	0	100	6220	172208	36491,68	134339	3878850	796066,95	17875	2830795	314434,66	100	100	100
4/5	100	0	100	7992	175932	81926,02	174469	4116104	1867197,71	22829	2890300	1129809,9	100	100	100
4/6	100	0	100	32022	819002	217613,3	695959	22181369	5498337,13	214194	34409076	5796728,02	100	100	100
5/1	100	0	100	5484	38916	28379,9	114921	802493	583865,11	15844	258876	184236,98	100	100	100
5/2	100	0	100	7486	39460	35413,44	157754	833872	752141,35	21398	262522	235807,81	100	100	100
5/3	100	0	100	30878	175944	51881,88	665082	3943076	1138903,05	205887	2893078	544105,1	100	100	100
5/5	100	0	100	35960	821376	239841,2	778103	22999202	5994550,08	239282	34488616	6370166,31	100	100	100
5/6	100	0	100	160220	3535910	667357,2	3820221	142336049	19334489,69	2642705	388934231	31556534,86	100	100	100
5/7	100	0	100	168766	3672988	1172415	4076055	149726077	38204901,91	2775260	403908847	83212859,51	100	100	100

6/1	100	0	100	32290	179464	146963,7	682638	3716195	3046922,4	215872	2952379	2373725,97	100	100	100
6/2	100	0	100	159240	180218	173027,3	3657114	4097899	3887902,94	2624207	2961431	2845785,48	100	100	100
6/4	100	0	100	163130	826736	317571,1	3819050	23398324	8101684,35	2687566	34714042	9914356,51	100	100	100
6/6	100	0	100	745864	3737264	1350338	20788663	148332323	44659606,47	31352186	410797175	104405463,9	100	100	100
6/7	100	0	100	784682	16447912	3552726	22024788	1076856964	138381263	32970049	4818682167	415945428,7	100	100	100
6/8	100	0	100	3642934	16604634	8509585	119867779	1101856437	472076361	400935595	4863223209	2074333822	100	100	100
7/1	100	0	100	156454	829848	721918	3387105	17920592	15272482,82	2583619	34857196	30084339,68	100	100	100
7/2	100	0	100	775844	836202	809050,1	19617829	22394729	21117884,25	32612535	35145063	34002090,74	100	100	100
7/3	100	0	100	779686	832230	812051,1	20757654	23337254	22214983,5	32825011	34952463	34126299,52	100	100	100
7/5	100	0	100	775638	3743032	2247848	21454713	146248309	80174608,98	32696640	411788350	219762752,9	100	100	100
7/7	100	0	100	3571036	16891650	9556575	123224696	1088316863	524027325,6	392828506	4948157090	2427670884	100	100	100
7/8	100	0	100	3714188	18886200	16636550	138022492	2029292649	1032858343	408599258	4955781120	4785334540	100	100	100
8/1	100	0	100	753538	3732166	3369590	18397369	81041204	73670616,91	31789989	410475184	367759506,3	100	100	100
8/2	100	0	100	3554754	3770884	3706693	114856850	138603217	126385837,7	391227759	414776764	407747307,9	100	100	100
8/6	100	0	100	3655268	16895356	13360511	132736611	1076755096	780943738,1	402089612	4948970194	3736786303	100	100	100

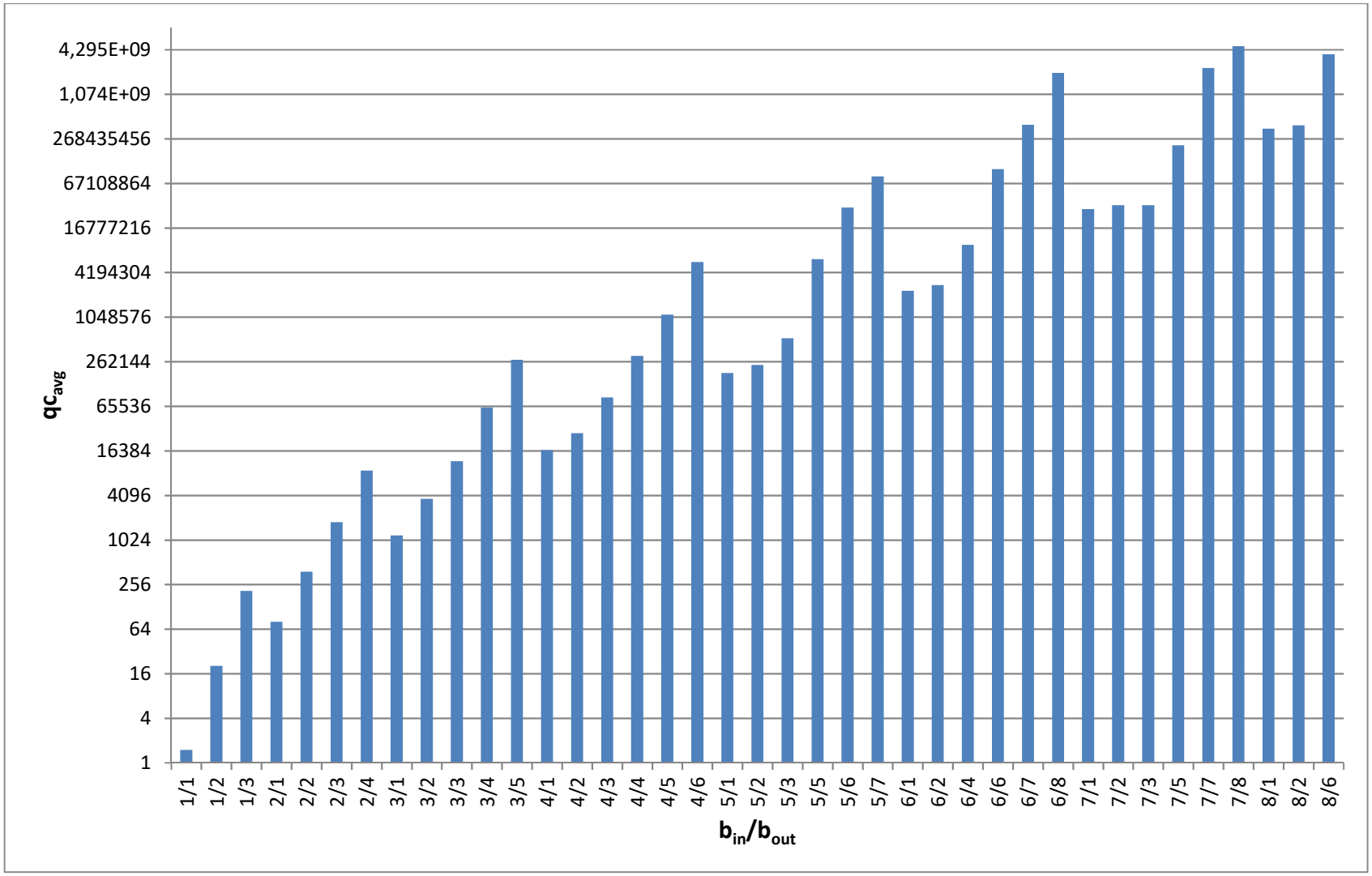
U ovome dijelu analizirat ću korake pretvorbe reverzibilnog sklopovlja u kvantno sklopovlje upotrebom CS dekompozicije, te generiranje QCL koda na temelju generiranog kvantnog sklopovlja. Sljedeći grafikon prikazuje prosječnu duljinu generiranog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 19: Prosječna duljina kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (CS dekompozicija)

S obzirom da dubina binarnog stabla CS algoritma raste u logaritamskoj ovisnosti o dimenziji unitarne matrice koja se dekomponira, tako i broj elemenata/matrica u binarnom stablu CS algoritma, koje će se u konačnici pretvoriti u kanonski oblik kontroliranih kvantnih vrata viših razina (slika 12), polinomno raste s dimenzijom unitarne matrice. Dimenzija unitarne matrice koja se dekomponira eksponencijalno ovisi o broju bitova u reverzibilnom sklopovlju, koje pak ovisi o broju ulaznih i izlaznih bitova (grafikon 8), tako da možemo zaključiti da duljina kvantnog sklopovlja eksponencijalno ovisi o broju ulaznih i izlaznih bitova VHDL programa, što je i vidljivo na prethodnom grafikonu. Također, možemo primijetiti i sličan trend između grafikona 8, na kojem je prikazan prosječni broj bitova reverzibilnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa, i grafikona 18, što potvrđuje prethodnu tvrdnju.

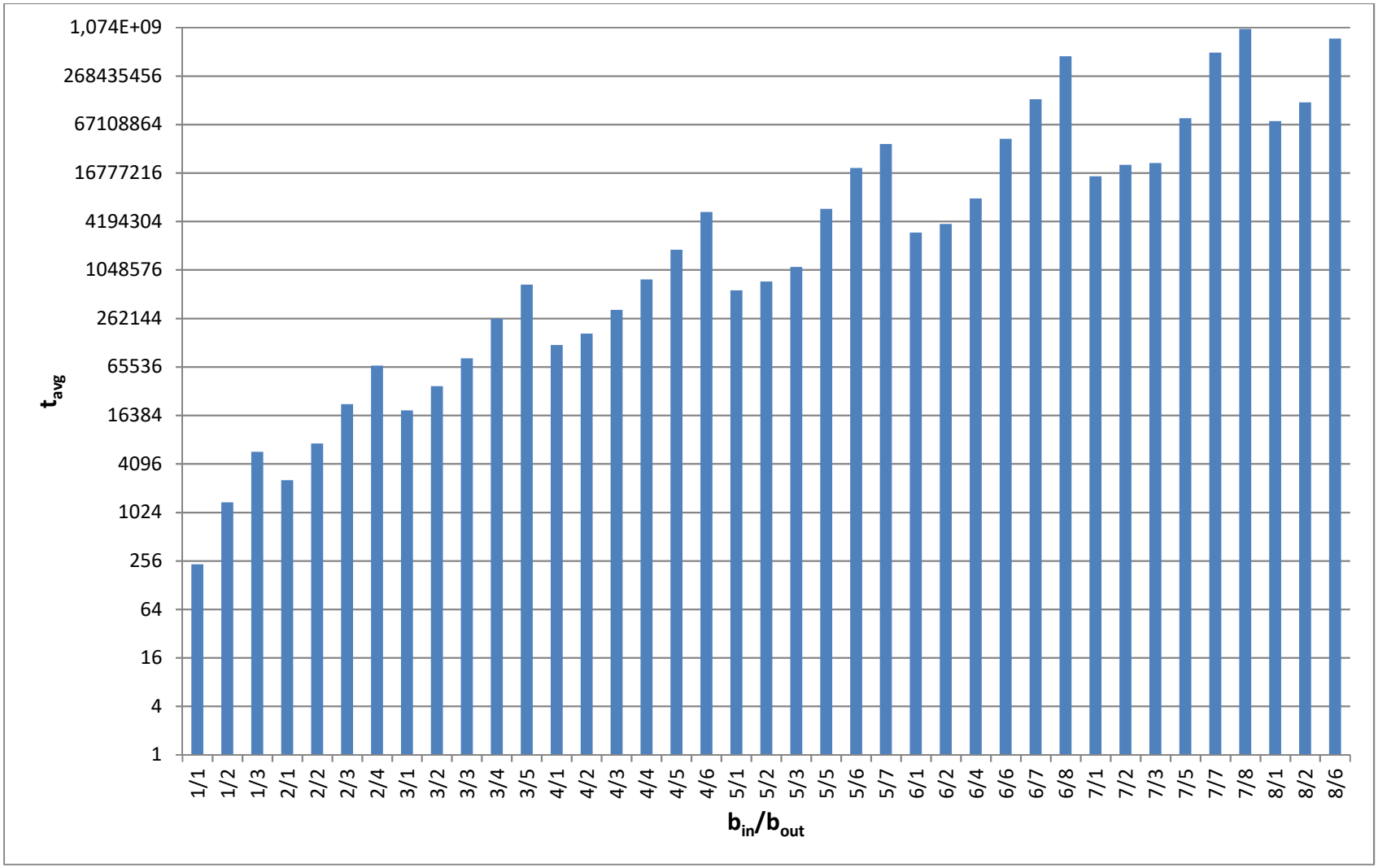
Sljedeći grafikon prikazuje prosječni kvantni trošak generiranog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa na logaritamskoj skali.



Grafikon 20: Prosječni kvantni trošak kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (CS dekompozicija)

Sve matrice binarnog stabla CS algoritma koje se pretvaraju u kvantna vrata (listovi stabla i centralne matrice) se u biti pretvaraju u kontrolirana kvantna vrata viših dimenzija. To znači da će kvantni trošak generiranog sklopovlja ovisiti polinomno o duljini samog kvantnog sklopovlja, na što ukazuje sličan trend prethodnog grafikona i grafikona 18. Kako duljina kvantnog sklopovlja koje je generirano CS algoritmom eksponencijalno ovisi o broju ulaznih i izlaznih bitova VHDL programa, možemo zaključiti da i kvantni trošak kvantnih sklopovlja koji su generirani CS algoritmom eksponencijalno ovisi o broju ulaznih i izlaznih bitova VHDL programa, što je i jasno vidljivo na prethodnom grafikonu. Kao i kod QR algoritma, broj testova u kojima generirano sklopovlje ima manji kvantni trošak ili kvantni trošak jednak reverzibilnom sklopovlju pada s porastom ulaznih odnosno izlaznih bitova. Također, iz kolona $N_{\leq rq}$ i $N_{>rqc}$ u tablici 19 vidljivo je da ovaj algoritam može generirati kvantno sklopovlje koje ima manji kvantni trošak ili kvantni trošak jednak kvantnom trošku reverzibilnog sklopovlja samo pri manjem broju ulaznih i izlaznih bitova.

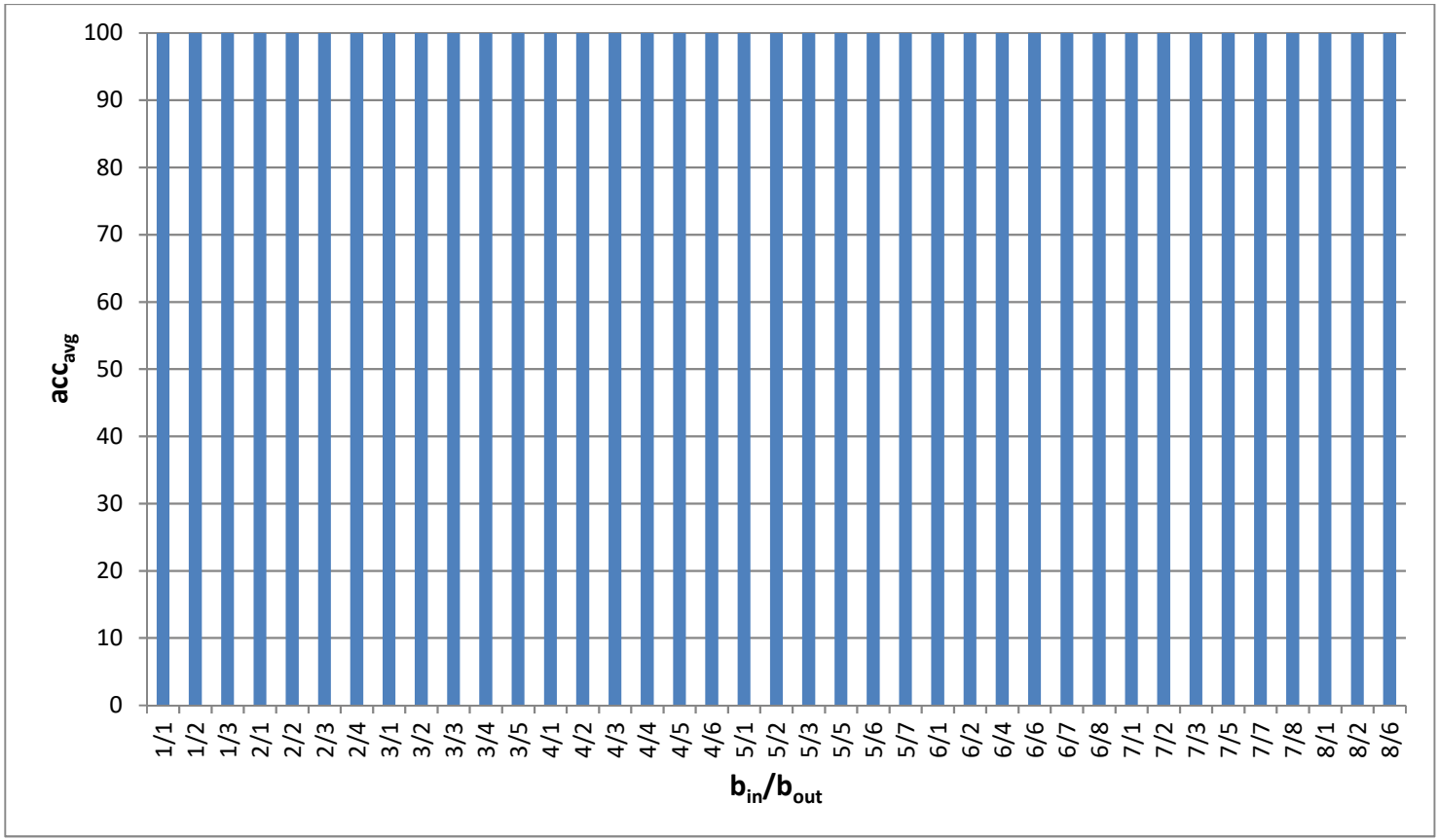
Sljedeći grafikon prikazuje prosječno vrijeme izvođenja preostalih koraka implementiranog modela (generiranje kvantnog sklopovlja na temelju reverzibilnog sklopovlja, te generiranje QCL koda na temelju kvantnog sklopovlja) upotrebom CS algoritma u odnosu na broj ulaznih i izlaznih bitova VHDL programa. Vrijednosti su prikazane na logaritamskoj skali.



Grafikon 21: Prosječno vrijeme trajanja generiranja kvantnog sklopovlja na temelju reverzibilnog sklopovlja, te generiranja QCL koda na temelju kvantnog sklopovlja u odnosu na broj ulaznih i izlaznih bitova VHDL programa (CS dekompozicija)

Pretvaranje reverzibilnog sklopovlja u kvantno sklopovlje ima dva bitna dijela, to su generiranje unitarne matrice reverzibilnog sklopovlja i dekompozicija te unitarne matrice CS algoritmom. Složenost generiranja unitarne matrice na temelju reverzibilnog sklopovlja ovisi o duljini reverzibilnog sklopovlja, te o broju bitova reverzibilnog sklopovlja. Unitarna matrica se generira tako da se sva reverzibilna vrata pretvaraju u unitarne matrice te se množe. Dakle, dimenzija unitarne matrice ovisit će eksponencijalno o broju bitova reverzibilnog sklopovlja, odnosno o broju ulaznih i izlaznih bitova VHDL programa (grafikon 8), te će duljina reverzibilnog sklopovlja isto tako eksponencijalno ovisiti o broju ulaznih i izlaznih bitova VHDL programa (grafikon 9). Složenost množenja kvadratnih matrica je polinomna u odnosu na dimenziju kvadratnih matrica, odnosno eksponencijalna u odnosu na broj ulaznih i izlaznih bitova VHDL programa. Dakle, složenost generiranja unitarne matrice reverzibilnog sklopovlja je eksponencijalno ovisna o broju ulaznih i izlaznih bitova VHDL programa. Zbog rekurzivnog načina funkcioniranja CS algoritma razumljivo je da će eksponencijalni rast dimenzije unitarne matrice prouzročiti dodatnu razinu u dubini binarnog stabla CS algoritma, što će značiti u prosjeku eksponencijalno više potrebnog vremena za izračun matrica na temelju kojih će se generirati kvantno sklopovlje. Dakako, složenost generiranja QCL koda na temelju kvantnog sklopovlja je polinomno ovisno o duljini kvantnog sklopovlja, što znači da je eksponencijalno ovisno o broju ulaznih i izlaznih bitova VHDL programa. Zaključujemo da je ukupna složenost opisana dva koraka eksponencijalna u odnosnu na broj ulaznih i izlaznih bitova VHDL programa, što je vidljivo na prethodnom grafikonu.

Sljedeći grafikon prikazuje prosječnu točnost generiranog QCL koda u odnosu na broj ulaznih i izlaznih bitova VHDL programa.



Grafikon 22: Prosječna točnost generiranog QCL izvornoga koda u odnosu na broj ulaznih i izlaznih bitova VHDL programa (CS dekompozicija)

Kao i kod algoritma izravne pretvorbe te QR algoritma, tako su i upotrebom CS algoritma sve instance testova uspješno prošle s gledišta točnosti generiranog QCL koda, dakle možemo zaključiti da prilikom pretvorbe VHDL koda u QCL kod nije došlo do pogreške zbog nužnog zaokruživanja koje je bilo postavljeno na 10 decimalnih mjesta.

8.4. Kritički osvrt

Sljedeća tablica prikazuje prosječna vremena izvođenja pojedine grupe testova, prosječni kvantni trošak i prosječnu točnost u pojedinoj grupi testova za svaki algoritam.

Tablica 23: Prosječna vremena trajanja pretvorbe VHDL koda u QCL kod i prosječni kvantni trošak za svaki algoritam

b_{in}/b_{out}	N_{test}	IP $N_{\leq rqc}$	QR $N_{\leq rqc}$	CS $N_{\leq rqc}$	IP $t_{avg}(\mu s)$	QR $t_{avg}(\mu s)$	CS $t_{avg}(\mu s)$	IP qc_{avg}	QR qc_{avg}	CS qc_{avg}	IP $acc_{avg}(\%)$	QR $acc_{avg}(\%)$	CS $acc_{avg}(\%)$
1/1	100	100	51	75	276597,45	276763,49	276696,76	0,49	2,48	1,49	100	100	100
1/2	100	100	43	9	600462,07	601074,82	601641,66	2,35	19,64	20,4	100	100	100
1/3	100	100	9	3	615284,46	616538,225	620809,48	17,02	117,59	210,42	100	100	100
2/1	100	100	12	12	290893,09	290754,55	292718,75	4,29	60,47	80,59	100	100	100
2/2	100	100	3	0	303784,99	304155,53	310267,9	22,39	270,52	379,92	100	100	100
2/3	100	100	2	0	643689,1	647044,68	665924,91	93,11	1049,92	1778,52	100	100	100
2/4	100	100	0	0	655743,65	665499,18	723291,28	333,65	3804,86	8839,35	100	100	100
3/1	100	100	0	0	311742,69	313768,15	329966,93	19,69	869,48	1180,63	100	100	100
3/2	100	100	0	0	332296,19	336377,46	368430,35	110,8	2435,1	3672,97	100	100	100
3/3	100	100	0	0	350553,43	362553,26	432957,9	373,69	6476,81	11873,67	100	100	100
3/4	100	100	0	0	694459,26	757912	953650,02	1475,15	30243,64	62922	100	100	100
3/5	100	100	0	0	724573,55	956780,24	1410663,95	4642,56	105005,39	277395,55	100	100	100
4/1	100	100	0	0	382653,56	403727,08	502768,84	86,01	12330,59	16802,67	100	100	100
4/2	100	100	0	0	532552	564418,29	701525,49	637,8	18497,33	28426,87	100	100	100
4/3	100	100	0	0	484524,36	580179,64	815994,64	1623,51	50398,64	85897,15	100	100	100
4/4	100	100	0	0	518922,42	829387,98	1313166,61	4867,45	155733,34	314434,66	100	100	100
4/5	100	100	0	0	817201,42	1859542,5	2681413,75	12815,37	480706,38	1129809,9	100	100	100
4/6	100	100	0	0	904156,67	5830003,86	6397479,5	41015,06	2079948,93	5796728,02	100	100	100
5/1	100	100	0	0	526696,28	732307,08	1110228,69	350,74	138772,52	184236,98	100	100	100
5/2	100	100	0	0	567919,07	861737,1	1317914,97	3498,24	165382,3	235807,81	100	100	100

5/3	100	100	0	0	662806,16	1317592,15	1798909,1	6781,02	336187,63	544105,1	100	100	100
5/5	100	100	0	0	831074,04	7901169,64	6820563,85	41475,87	2977713,66	6370166,31	100	100	100
5/6	100	100	0	0	1346533,94	30910996,33	20670823,95	138466,5	12490198,9	31556534,9	100	100	100
5/7	100	100	0	0	1605744,47	80591638,9	39795545,57	265839,6	28105698	83212859,5	100	100	100
6/1	100	100	0	0	639597,65	3950822,61	3685981,96	1514,08	1777326,69	2373725,97	100	100	100
6/2	100	100	0	0	930433,23	4854465,7	4814917,86	21323,65	2013098,75	2845785,48	100	100	100
6/4	100	100	0	0	1266347,72	14353785,02	9361209,54	55809,33	5940671,02	9914356,51	100	100	100
6/6	100	100	0	0	2267461,44	140604558	46909402,22	299748,9	46690340,6	104405464	100	100	100
6/7	100	100	0	0	3885361,11	489211380,8	142234341,8	867081,2	164708268	415945429	100	100	100
6/8	100	100	0	0	6007051,03	1566637453	478022859,2	2374256	634353256	2074333822	100	100	100
7/1	100	100	0	0	922405,8	46307391,09	16193882,28	6107,81	22541811,6	30084339,7	100	100	100
7/2	100	100	0	0	2529345,69	54163435,6	23638464,73	132838,8	23877882,8	34002090,7	100	100	100
7/3	100	100	0	0	2899405,58	59936274,52	25102138,91	150945,1	24280800,4	34126299,5	100	100	100
7/5	100	100	0	0	4947929,32	382881746,3	85099857,25	515226,9	124935880	219762753	100	100	100
7/7	100	100	0	0	11003868,18	2884895105	534967188,5	2630831	1028359457	2427670884	100	100	100
7/8	100	100	0	0	17160478,26	4381483918	1049903904	5266604	1992623032	4785334540	100	100	100
8/1	100	100	0	0	2135557,71	684503034,2	75802818,92	25984,76	270866515	367759506	100	100	100
8/2	100	100	0	0	10208695,17	711480896,7	136568411,4	775404	288707046	407747308	100	100	100
8/6	100	100	0	0	26399279,26	2870659799	807257497,7	3796801	1959042459	3736786303	100	100	100

Prosječno vrijeme trajanja u grupama testova u kojima se koristio izravni algoritam za prevođenje reverzibilnog sklopovlja u kvantno prevođenja VHDL koda u QCL je najmanje u odnosu na prosječna vremena QR i CS algoritama. Ono što je zanimljivo za primijetiti je da su se grupe testova koje su koristile QR algoritam u prosjeku brže izvršavale od grupe testova u kojima se koristio CS algoritam sve dok je broj $max(b_{in}, b_{out}) < 7$, nakon što je broj $max(b_{in}, b_{out}) > 6$, CS algoritam postaje brži u odnosu na QR algoritam. Iznimka je grupa testova u kojoj je $b_{in} = 4$ i $b_{out} = 6$, u kojoj je QR algoritam ipak nešto brži od CS algoritma. U svim provedenim testovima primjetno je da ukupno vrijeme prevođenja raste eksponencijalno u odnosu na broj ulaznih bitova bez obzira koji algoritam je korišten za pretvorbu reverzibilnog sklopovlja u kvantno. Iz prikazanih vremena prevođenja koja eksponencijalno rastu u odnosu na ulazne/izlazne bitove možemo zaključiti da će pomoću implementiranog prevoditelja biti moguće provesti prevođenje na relativnom manjem broju ulaznih/izlaznih bitova (do 16-ak) u prihvatljivom vremenu (unutar 2 sata) na računalu s 4 GB RAM-a i i5 2.3 GHz-nim procesorom na Linux Ubuntu 13.10 operacijskom sustavu.

Kvantna sklopovlja koja su generirana u grupama testova u kojima se koristio izravni algoritam za prevođenje reverzibilnog sklopovlja u kvantno imaju najmanji prosječni kvantni trošak u odnosu na kvantni trošak sklopovlja koja su generirana u grupama testova koja su koristila QR i CS algoritme. Testovi u kojima se koristio QR algoritam su generirala kvantna sklopovlja koja u prosjeku imaju manji kvantni trošak od sklopovlja koja su generirana u grupama testova u kojima se koristio CS algoritam. Iznimka su prve dvije grupe testova u kojima je prosječni kvantni trošak kvantnih sklopovlja koja su generirana CS algoritmom nešto manja od prosječnog kvantnoga troška kvantnih sklopovlja koja su generirana QR algoritmom. Također, vidimo da u slučaju QR i CS algoritama broj testova u kojima je generirano kvantno sklopovlje koje ima manji kvantni trošak ili kvantni trošak jednak reverzibilnom sklopovlju drastično pada s brojem bitova reverzibilnog sklopovlja, odnosno dimenzijom unitarne matrice. Kod algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, kvantni trošak generiranog sklopovlja je uvijek jednak kvantnom trošku reverzibilnog sklopovlja.

U provedenim testiranjima koja su opisana u poglavljima 7 i 8 zaključili smo da je kvantni trošak generiranog sklopovlja kod algoritma za izravnu pretvorbu reverzibilnog sklopovlja u kvantno uvijek jednak kvantnom trošku reverzibilnog sklopovlja. Implementirani prevoditelj u procesu prevođenja koristi algoritam za izravnu pretvorbu reverzibilnog sklopovlja u kvantno, stoga možemo potvrditi podhipotezu **H1A** te možemo zaključiti da je

na klasičnom računalu moguće implementirati automatskog prevoditelja izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala, pri čemu će kvantni trošak generiranog sklopovlja biti manji ili jednak reverzibilnom sklopovlju koje je generirano na temelju izvornoga koda za programski jezik klasičnoga računala.

Prosječna točnost pretvaranja VHDL koda u QCL kod se pokazala maksimalnom u svim instancama testova implementiranog prevoditelja, stoga možemo potvrditi hipotezu **H1B**, odnosno možemo tvrditi da je moguće uspješno implementirati prevoditelja koji će izvorni kod klasičnih računala prevoditi u izvorni kod kvantnih računala sa 100% točnošću. Isto tako, vidjeli smo da algoritam za izravno pretvaranje reverzibilnog sklopovlja u kvantno koristi cjelobrojnu aritmetiku stoga možemo zaključiti da će taj algoritam osigurati 100% točnost pretvorbe i u složenijim slučajevima pretvorbe izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala.

Nakon što smo potvrdili podhipoteze **H1A** i **H1B,** možemo u konačnici potvrditi hipotezu **H1** i tvrditi da je na klasičnom računalu moguće uspješno implementirati automatsko prevođenje izvornoga koda za programske jezike klasičnoga računala u izvorni kod za programske jezike kvantnoga računala, međutim uz implementacijska ograničenja koja su navedena u poglavlju 6.2.2.

9. Buduća istraživanja

Jedan od predmeta budućih istraživanja bit će unaprjeđenje opisanog modela za pretvorbu izvornoga koda klasičnih računala u izvorni kod kvantnih računala. Trenutno opisani model pretvara izvorni VHDL kod u tablicu istinitosti, koju zatim proširuje tako da bude reverzibilna. Na temelju reverzibilne tablice istinitosti generira se reverzibilno sklopovlje (korak 1 na slici 31). Ideja je da se taj korak zamijeni na način da se pronađu uzorci dizajna [35] u izvornom kodu klasičnih računala te da se onda isti pretvore u dizajne uzoraka za reverzibilna računala te se na taj način eliminiraju implementacijska ograničenja predloženog pristupa u ovom radu. Smatram da je ovo vrlo bitan korak u budućim istraživanjima zato jer će na taj način biti moguće iskoristiti kvantnomehaničke efekte poput superpozicije za brže pronalaženje rješenja. Također, jedno od mogućih poboljšanja modela je uvođenje koraka optimizacije u sam model. U tom novom koraku bi se generirano kvantno sklopovlje optimiziralo jednim od algoritama za optimizaciju kvantnog sklopovlja poput Banerjeeovog i Pathakovog algoritma [4], Sasanianovog i Millerovog algoritma [89], Abdessaiedovog, Soekenovog i Drechslerovog algoritma [1], Sedlakovog i Pleshovog algoritma [93], Da Silvinog, Piusovog i Kashefiovog algoritma [18], koji za cilj imaju smanjiti kvantni trošak ulaznog kvantnog sklopovlja.

Osim samog poboljšanja modela, predmetom istraživanja u budućnosti je i implementacija opisanih promjena modela u postojeći prevoditelj. Isto tako, jedan od ciljeva budućih istraživanja je i iskorištavanje implementiranog okvira za analizu drugih algoritama za generiranje kvantnih sklopova poput Daskininovog i Kaisovog algoritma [19], Lukacovog i Perkowskiovog algoritma [74] te Yabukievog i Ibaevog algoritma [114]. Također, cilj u budućim istraživanjima je i testiranje prevoditelja nad drugim izvorima VHDL koda, te nad programskim jezicima klasičnih računala za koje već postoji prevoditelj u VHDL izvorni kod. Jedan od zadataka budućih istraživanja je i proširivanje funkcionalnosti implementiranog prevoditelja u smislu omogućavanja pretvaranja izvornoga koda klasičnih računala u izvorni kod drugih programskih jezika za kvantna računala osim QCL-a. Pored toga, jedna od zadaća u budućnosti je i optimizacija trenutno implementiranog prevoditelja u smislu ubrzavanja izvođenja pojedinih koraka, te optimizacija korištene memorije. Neki dijelovi implementiranog prevoditelja, poput pretvaranja kvantnog sklopovlja u QCL kod, bit će prilagođeni i ponuđeni autorima Revkit biblioteke [54], koja se i koristila prilikom implementacije samog prevoditelja.

10. ZAKLJUČAK

U ovom radu predstavljen je model za pretvaranje izvornoga koda klasičnih računala u izvorni kod kvantnih računala. Motivaciju za razvijanje jednog takvog modela sam pronašao ponajviše u tome da je kvantno računalo mogući nasljednik klasičnoga računala. Ukoliko kvantno računalo postane nasljednik klasičnoga računala, postojeće programe klasičnih računala će biti potrebno realizirati i na samim kvantnim računalima. Opisani model automatizira proces prevođenja izvornoga koda klasičnoga računala u izvorni kod kvantnoga računala, pa tako može olakšati i ubrzati razvoj programa za kvantna računala koja se temelje na programima za klasična računala. Osim samog modela, rezultat ovog istraživanja je i prevoditelj koji je implementiran na temelju opisanog modela. Dokazano je da je moguće uz određena ograničenja implementirati, odnosno u potpunosti automatizirati proces prevođenja izvornoga VHDL koda u QCL kod. Dokazano je da je prosječna točnost generiranog izvornoga koda za kvantna računala 100%, što znači da nije dolazilo do grešaka zbog neminovnog „zaokruživanja“ u samoj implementaciji. Upotrebu implementiranog prevoditelja u budućnosti vidim u istraživačkom radu i edukaciji. U provedenom istraživanju implementirani prevoditelj poslužio je i u svrhu sagledavanja odnosa između QR i CS algoritama po pogledu utrošenog vremena i generiranog kvantnoga troška kvantnih sklopova koji su generirani na temelju skupa unitarnih matrica koje reprezentiraju logiku programa za klasična računala. Pokazano je da na spomenutom skupu podataka CS algoritam u prosjeku brži od QR algoritma, međutim sklopovlja koja su generirana CS algoritmom imaju i veći prosječni kvantni trošak. Smatram da je to saznanje vrlo bitno zato što se u literaturi CS algoritam spominje kao algoritam koji generira kraće kvantno sklopovlje od QR algoritma [95], što dakako može rezultirati i manjim kvantnim troškom, međutim to nad spomenutim skupom podataka u sklopu ovog istraživanja nije bio slučaj. Također, treba napomenuti da su oba algoritma u velikoj većini slučajeva generirala veći kvantni trošak od kvantnoga troška generiranog reverzibilnog sklopovlja, što znači da se nisu postigla nikakva poboljšanja u smislu brzine izvršavanja pri samom prevođenju izvornoga koda klasičnih računala u izvorni kod kvantnih računala. Isto tako, vrlo bitno saznanje koje je proizašlo iz istraživanja je to da je Millerov, Maslov i Dueckov algoritam u sinergiji s izravnim algoritmom za pretvorbu reverzibilnog sklopovlja u kvantno sklopovlje značajno bolja opcija za prevođenje izvornoga koda klasičnih računala u izvorni kod kvantnih računala od QR i CS algoritama po pitanju kvantnoga troška i utrošenog vremena potrebnog za prevođenje.

LITERATURA

[1] Abdessaied, N.; Soeken, M.; Drechsler, R. Quantum Circuit Optimization by Hadamard Gate Reduction, *Lecture Notes in Computer Science*, Vol. 8507, pages 149 - 162, 2014.

[2] Arora, S.; Barak B. *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, 2009.

[3] Ashenden, P.J. The Designer's Guide to VHDL, *Morgan Kaufmann Publishers Inc.*, San Francisco, 2008.

[4] Banerjee, A.; Pathak, A. An Algorithm for Minimization of Quantum Cost, *Applied Mathematics & Information Sciences*, Vol 6., No. 1, 2012.

[5] Bell, J. S. On the Einstein-Podolsy-Rosen paradox, *Physics*, volume 1, pages 195–200, 1964.

[6] Boixo, S.; Rønnow, T. F.; Isakov, S. V.; Wang, Z.; Wecker, D.; Lidar, D. A.; Martinis, J. M.; Troyer, M. Quantum annealing with more than one hundred qubits, *Nature Physics*, Vol. 10, 2014.

[7] Bojić, A. A New Quantum Game Based on CHSH Game, *Journal of Information and Organizational Sciences*, Vol. 37, No. 1, 2013.

[8] Bojić, A. An approach to source code conversion of classical programming languages into source code of quantum programming languages, *Journal of Information and Organizational Sciences*, Vol. 38, No. 2, 2014.

[9] Bojić, A. Quantum Algorithm for Finding a Maximum Clique in an Undirected Graph, *Journal of Information and Organizational Sciences*, Vol. 36, No. 2, 2012.

[10] Boyer, M.; Brassard, G.; Høyer, P.; Tapp, A. Tight bounds on quantum searching. *Fortschritte der Physik*, Vol. 46, pages 493 - 505, 1998.

[11] Brandhorst-Satzkorn, J. A Review of Freely Available Quantum Computer Simulation Software, Linköping University, 2012.

[12] Brandt, H. Quantum Computer Circuit Analysis and Design, Army Research Laboratory, USA, 2009.

[13] Briegel, H.J.; Calarco, T.; Jaksch, D.; Cirac, J.I.; Zoller, P. Quantum computing with neutral atoms, *Journal of Modern Optics* , Vol. 47, pages 415 – 451, 2000.

[14] Caraiman, S.; Manta, V. Parallel Simulation of Quantum Search, *International Journal of Computers - Communications & Control,* Vol. 5, pages 634 - 641, 2010.

[15] Chen, Y. G.; Wang, J. B. Qcompiler: quantum compilation with CSD method, *Computer Physics Communications*, Vol. 184, Issue 3, pages 853 - 865, 2013.

[16] Clauser, F.; Horne, M. A.; Shimony, A.; Holt, R. A. Proposed Experiment to Test Local Hidden-Variable Theories, *Physical Review Letters,* volume 23, pages 880-884, 1969.

[17] Cybenko, G. Reducing quantum computations to elementary unitary operations, Computing in Science & Engineering, Vol. 3, pages 27 -32, 2001.

[18] Da Silva, R. D., Pius, E., Kashefi, E. Global Quantum Circuit Optimization, URL http://arxiv.org/pdf/1301.0351v1, učitano 09.05.2014.

[19] Daskin, A.; Kais S. Decomposition of unitary matrices for finding quantum circuits: Application to molecular Hamiltonians, *The Journal of Chemical Physics,* Vol. 134, 2011.

[20] De Raedt H.; Michielsen K. Computational Methods for Simulating Quantum Computers, *Handbook of Theoretical and Computational Nanotechnology*, 2004.

[21] Deutsch D.; Jozsa R. Rapid solutions of problems by quantum computation, In *Proceedings of the Royal Society of London A 439,* pages 553 - 558, 1992.

[22] Deutsch, D. Quantum theory, the Church-Turing principle and the universal quantum computer, In *Proceedings of the Royal Society of London A 400*, pages 97 - 117, 1985.

[23] Deutsch, I.H.; Brennen, G.K.; Jessen, P.S. Quantum computing with neutral atoms in an optical lattice, *Fortschritte der Physik*, Vol. 48, pages 925 – 943, 2000.

[24] DiCarlo, L.; Chow, J. M.; Gambetta, J. M.; Bishop L. S.; Johnson, B. R.; Schuster, D. I.; Majer, J.; Blais, A.; Frunzio, L.; Girvin, S. M.; Schoelkopf, R. J. Demonstration of two-qubit algorithms with a superconducting quantum processor, *Nature*, Vol. 460, 2009.

[25] DiVincenzo, D. P. The Physical Implementation of Quantum Computation, *Fortschritte der Physik*, Vol. 48, Issue 9 – 11, pages 771–783, 2000.

[26] Divjak, B.; Lovrencic, A. *Diskretna matematika s teorijom grafova*, TIVA Tiskara Varaždin, 2005.

[27] Drechsler, R.; Wille R. From Truth Tables to Programming Languages: Progress in the Design of Reversible Circuits, *International Symposia on Multiple-Valued Logic*, pages 78 - 85, 2011.

[28] Duan, L.M.; Guo, G.C. Reducing decoherence in quantum-computer memory with all quantum bits coupling to the same environment, *Physical Review A*, Vol. 57, 1998.

[29] Einstein, A.; Podolsky, B.; Rosen N. Can Quantum-Mechanical Description of Physical Reality be Considered Complete, *Physical Review*, volume 47, pages 777-780, 1935.

[30] Englert, B. G. U. Mesoscopic Shelving Readout of Superconducting Qubits in Circuit Quantum Electrodynamics, Diploma Thesis, Bavarian Academy of Sciences and Humanities, 2008.

[31] Feynman, R. P. Simulating Physics with Computers, *International Journal of Theoretical Physics,* Vol. 21, No. 6/7, pages 467 - 488, 1982.

[32] Franco, R. Grover's algorithm and human memory, 2008, URL http://arxiv.org/pdf/0804.3294v1.pdf

[33] Fuechsle, M.; Miwa, J.A.; Mahapatra, S.; Ryu, H.; Lee, S.; Warschkow, O.; Hollenberg, L.C.L.; Klimeck, G.; Simmons, M.Y. A single-atom transistor, *Nature Nanotechnology*, Vol. 7, pages 242 – 246, 2012.

[34] Gadner, W. Algorithms for the QR-Decomposition, *Research report 80-021*, Angewandte Mathematik Eidgenoessische Technische Hochschule, Zuerich, 1980.

[35] Gamma, E. Design patterns : elements of reusable object-oriented software, Addison-Wesley, 1995.

[36] Giovannetti, V.; Lloyd, S.; Maccone, L. Quantum random access memory, *Physical Review Letters*, Vol. 100, 2008.

[37] Grattage, J. An overview of QML with a concrete implementation in Haskell, *Proceedings of Quantum Physics and Logic*, pages 157 - 165, 2008.

[38] Grover, L.K. A fast quantum mechanical algorithm for database search. *28th Annual ACM Symposium on the Theory of Computing*, pages 212 - 219, Philadelphia, USA, 1996.

[39] Gupta, S. User Manual for the SPARK Parallelizing High-Level Synthesis Framework, The Regents of the University of California., 2004.

[40] Hamann, S. E. Physical Review Letters, Vol. 80, 1998.

[41] http://1qbit.com/, učitano 11.02.2015.

[42] http://en.wikipedia.org/wiki/Quantum_complexity_theory, učitano 1.12.2012.

[43] http://en.wikipedia.org/wiki/Quantum_computer, učitano 12.03.2014

[44] http://en.wikipedia.org/wiki/Quantum_decoherence, učitano 31.08.2014.

[45] http://en.wikipedia.org/wiki/Quantum_mechanics, učitano 23.10.2012

[46] http://en.wikipedia.org/wiki/Toffoli_gate, učitano 28.12.2012.

[47] http://googleresearch.blogspot.co.uk/2013/05/launching-quantum-artificial.html, učitano 19.08.2014.

[48] http://mesl.ucsd.edu/spark/, učitano 01.11.2013.

[49] http://ns.umich.edu/index.html?Releases/2005/Dec05/r121205b, učitano 02.02.2014

[50] http://plato.stanford.edu/entries/qt-entangle/, učitano 27.11.2012

[51] http://scriptsharp.com/, učitano 02.10.2014.

[52] http://www.dwavesys.com/quantum-computing/applications, učitano 25.11.2014

[53] http://www.extremetech.com/computing/84228-first-ever-commercial-quantum-computer-now-available-for-10-million, učitano 23.05.2014

[54] http://www.informatik.uni-bremen.de/revkit/, učitano 03.07.2013.

[55] http://www.libquantum.de/, učitano 01.06.2016.

[56] http://www.netlib.org/lapack/, učitano 22.08.2013.

[57] http://www.p3cobol.com/, učitano 02.10.2014.

[58] http://www.quantiki.org/wiki/List_of_QC_simulators, učitano 01.03.2013

[59] http://www.washingtonpost.com/world/national-security/nsa-seeks-to-build-quantum-computer-that-could-crack-most-types-of-encryption/2014/01/02/8fff297e-7195-11e3-8def-a33011492df2_story.html, učitano 07.09.2014.

[60] https://code.google.com/p/j2objc/, učitano 01.10.2014.

[61] https://docs.microsoft.com/en-us/quantum/?view=qsharp-preview, učitano 26.03.0218.

[62] https://en.wikipedia.org/wiki/Simon%27s_problem

[63] https://gna.org/projects/ghdl/, učitano 03.07.2014.

[64] https://hr.wikipedia.org/wiki/Kvantna_mehanika, učitano 27.11.2015.

[65] https://www.nextplatform.com/2015/11/09/the-future-of-quantum-computing-will-be-hybrid, učitano 26.03.2018.

[66] https://www.rigetti.com/forest, učitano 26.03.2018.

[67] JavadiAbhari, A. Scaffold: Quantum Programming Language, Technical Report TR-934-12, Princeton University, USA, 2012.

[68] JavadiAbhari, A.; Patil, S.; Kudrow, D.; Heckey, J.; Lvov, A.; Chong, F.T.; Martonosi, M. ScaffCC: Scalable Compilation and Analysis of Quantum Programs, *Journal of Parallel Computing*, Vol. 45, 2015.

[69] Jiayu, Z.; Junsuo, Z.; Fanjiang, X.; Haiying, H.; Peng, Q. Analysis and Simulation of Grover's Search Algorithm, International *Journal of Machine Learning and Computing*, Vol. 4, No. 1, 2014.

[70] Kaye, P.; Laflamme, R.; Mosca, M. *An Introduction to Quantum Computing*, Oxford University Press, USA, 2007.

[71] Lavor, C.; Liberti, L.; Maculan, N. Grover's algorithm applied to the molecular distance geometry problem, In *Proceedings of VII Brazilian Congress of Neural Networks,* 2005.

[72] Lee, M.; Tse, D.; Goldburg, W. I.; Lowe, I. J. Nuclear-Magnetic-Resonance Free Induction Decay in a Two-Spin System, *Physical Review*, Vol. 158, 1967.

[73] Loke, T.; Wang, J. B.; Chen, Y. H. OptQC: An optimized parallel quantum compiler, *Computer Physics Communications*, Vol 185, Issue 12, pages 3307 - 3316, 2014.

[74] Lukac, M.; Perkowski, M. Evolving Quantum Circuits using Genetic Algorithm, Evolvable Hardware, *NASA/DoD Conference*, 2002.

[75] Mano, M. M.; Kime C.R.; Logic and Computer Design Fundamentals: 4th Edition, Prentice Hall, 2007.

[76] Mateus, P.; Omar, Y. Quantum Algorithm for Closest Pattern Matching, *Quantum Information Processing*, IOS Press, 2006.

[77] Miller, D. M.; Maslov, D.; Dueck, G. W. A transformation based algorithm for reversible logic synthesis, *Proceedings of Design Automation Conference*, pages 318 - 323, 2003.

[78] Miszczak, J.A. Models of quantum computation and quantum programming languages, *Bulletin of Polish Academy of Sciences: Technical Sciences*, Vol. 59, No. 3, 2011

[79] Mlnarık, H. Quantum Programming Language LanQ, Masaryk University - Faculty of Informatics, Czech Republic, 2007.

[80] Nagarajan, R.; Papanikolaou, N. Simulating and Compiling Code for the Sequential Quantum Random Access Machine, *Electronic Notes in Theoretical Computer Science*, Vol. 170, pages 101 - 124, 2005

[81] Nakahara, M.; Ohmi, T. Quantum Computing: *From Linear Algebra to Physical Realizations*, Taylor & Francis Group, 2008.

[82] Neutral Atom Approaches to Quantum Information Processing and Quantum Computing: A Quantum Information Science and Technology Roadmap, *Advanced Research and Development Activity*, USA, 2004.

[83] Nielsen, Michael A.; Chuang, Isaac L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press, New York, 2011.

[84] Ömer, Bernhard. Structured Quantum Programming, *Institute for Theoretical Physics*, Rev. 2, Austria, 2009.

[85] Ozhigov, Y. Quantum Computers Speed Up Classical with Probability Zero, *Chaos Solitons Fractals*, Vol. 10, pages 1707 – 1714, 1999.

[86] Pittenger, A. O. An Introduction to Quantum Computing Algorithms, Birkhäuser Boston, 1999.

[87] Politi, A.; Matthews, J. C. F.; O'Brien, J. L. Shor's Quantum Factoring Algorithm on a Photonic Chip, *Science*, Vol. 325, 2009.

[88] Prechtel, J. H.; Kuhlmann, A.V.; Houel, J.; Ludwig, A.; Valentin, S.R.; Wieck, A. D.; Warburton, R. J. Decoupling a hole spin qubit from the nuclear spins, *Nature Materials*, Vol. 15, pages 981 - 986, 2016.

[89] Sasanian, Z.; Miller, D. M. Reversible and Quantum Circuit Optimization: A Functional Approach, *Lecture Notes in Computer Science*, Vol. 7581, pages 112 - 124, 2013.

[90] Schneider, S. Quantum System Simulator, *MIT*, USA, 2000.

[91] Schrödinger, E.; An Undulatory Theory of the Mechanics of Atoms and Molecules, *Physical Review*, Volume 28(6), pages 1049 - 1070, 1926.

[92] Smith, R.S.; Curtis, M. J.; Zeng, W. J. A Practical Quantum Instruction Set Architecture, Rigetti Computing, 2016.

[93] Sedlak, M.; Plesch, M. Towards optimization of quantum circuits, *Central European Journal of Physics*, Vol. 6, No. 1, pages 128 - 134, 2008.

[94] Selinger, P. Towards a quantum programming language, *Mathematical Structures in Computer Science*, Vol. 14, pages 527 - 586, 2004.

[95] Shende, V.V.; Bullock, S. S.; Markov, I. L. Synthesis of Quantum Logic Circuits, *IEEE Transactions on Computer-Aided Design*, Vol. 25, No. 6, pages 1000 - 1010, 2006.

[96] Shor, P. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Journal on Computing*, Volume 26, No. 5, pages 1484-1509, 1997.

[97] Simon, D.R; On the power of quantum computation, *Foundations of Computer Science*, 35th Annual Symposium, pages 116 - 123, 1994.

[98] Sofge, D. A Survey of Quantum Programming Languages: History, Methods, and Tools, *Proceedings of the Second International Conference on Quantum, Nano, and Micro Technologies*, IEEE Computer Society, pages 66 - 71, 2008.

[99] Steane, A.M. The Ion Trap Quantum Information Processor, *Applied Physics*, Vol. B64, 1997.

[100] Tucci, R.R. Qubiter Algorithm Modification, Expressing Unstructured Unitary Matrices with Fewer CNOTs, arXiv:quant-ph/0411027, 2004.

[101] Tucci, R.R., What is the CS Decomposition, and how does one use it to do quantum compiling?, URL http://www.ar-tiste.com/csd-intro.pdf, učitano 9.8.2014.

[102] Turing, A. On Computable Numbers, with an Application to the Entscheidungsproblem, *Proceedings London Mathematical Society*, Vol. 42, No. 1, pages 230 - 265, 1936.

[103] Udrescu, M.; Prodan, L.; Vladutiu, M. Implementing Quantum Genetic Algorithms: A Solution Based on Grover's Algorithm. In *Proceedings of the 3rd Conference on Computing Frontiers*, pages 71 - 82, 2006.

[104] Van der Sar, T.; Wang, Z. H.; Blok, M. S.; Bernien, H.; Taminiau, T. H.; Toyli, D. M.; Lidar, D. A.; Awschalom, D. D.; Hanson R.; Dobrovitski, V. V.;Decoherence-protected quantum gates for a hybrid solid-state spin register, *Nature*, Vol. 484, pages 82 - 86, 2012

[105] Van Loan, C. Computing the CS and the Generalized Singular Value Decompositions, *Numerische Mathematik*, Vol. 46, pages 479 - 491, 1985.

[106] Vandersypen, M. K.; Steffen, M.; Breyta, G.; Yannoni, C. S.; Sherwood, M. H.; Chuang, I. L. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance, *Nature*, Vol. 414, 2001.

[107] Vijayalakshmi B. Simulation of quantum circuits using QCAD, *International Journal of Engineering Research*, Vol. 10, 2015.

[108] Virginia, A.J.; Yankova, Y.D.; Bertel, K. L. M. An empirical comparison of ANSI-C to VHDL compilers: SPARK, ROCCC and DWARV, *Delft University of Technology*, 2007.

[109] Wang, Y. Quantum Computation and Quantum Information, *Statistical Science*, Vol. 27, No. 3, pages 373 - 394, 2012.

[110] Watrous, J. Quantum Computational Complexity, *Encyclopedia of Complexity and System Science*, Springer, 2009.

[111] Wille, R.; Saeedi, M.; Drechsler, R. Synthesis of Reversible Functions Beyond Gate Count and Quantum Cost, *International Workshop on Logic Synthesis*, 2009.

[112] Wille, R.; Soeken, M.; Drechsler, R. Introduction to Reversible and Quantum Circuits, *Springer*, 2015.

[113] Wineland, D. J.; Bergquist, J. C.; Bollinger, J. J.; Itano, W. M. Quantum Effects in Measurements on Trapped Ions, *Physica Scripta*, Vol. T59, pages 286 - 293, 1995.

[114] Yabuki, T.; Iba, H; Genetic Algorithms for Quantum Circuit Design - Evolving a Simpler Teleportation Circuit, *Genetic and Evolutionary Computation Conference*, 2000.

[115] Zalka, C. Grover's quantum searching algorithm is optimal, *Physical Review A*, Vol. 60, No. 4, pages 2746 - 2751, 1999.

ŽIVOTOPIS

Alan Bojić je rođen 07. studenog 1984. godine u Zagrebu. Osnovnu školu te 10. matematičku gimnaziju je završio u Zagrebu. Diplomirao je na Fakultetu organizacije i informatike u Varaždinu na temu: "Algoritmi i strukture podataka u Prologu". Do završetka studija aktivno se bavio vaterpolom te je igrao 1. Hrvatsku vaterpolo ligu za V.K. Medveščak iz Zagreba. Također, bio je i član juniorske hrvatske reprezentacije. Trenutno radi u Agramsoft softverskoj tvrtki kao arhitekt programskih rješenja. Upoznat je sa Java, PL/SQL, Oracle Forms, Oracle Reports, Oracle ADF, php, XML i SOA tehnologijama iz kojih posjeduje stručne certifikate. Istraživački interes mu je vezan uz teoriju računarstva i teoriju složenosti.

Obrazovanje (kronološki od novijeg k starijem datumu):	2003. - 2008. Fakultet organizacije i informatike, Sveučilište u Zagrebu, Pavlinska 2, Varaždin 1999. - 2003. 10. Gimnazija, Prirodoslovno-matematički smjer, Klaićeva 7, Zagreb, Hrvatska 1991. - 1999. Osnovna škola Petar Zrinski, Krajiška 9, Zagreb, Hrvatska
Radno iskustvo (kronološki od novijeg k starijem datumu):	2017. - 2018. Agramsoft Zagreb, Arhitekt programskih rješenja Projekti: • Oktal Pharma d.o.o. – Implementacija eCommerce rješenja • Oktal Pharma d.o.o. – Implementacija CMS rješenja 2014. - 2017. IN2 d.o.o. Zagreb, Voditelj Oracle razvoja Projekti: • Atlantic Group d.d. - Vođenje razvoja i održavanja Opus*Erp-a • Oktal Pharma d.o.o. - Vođenje razvoja i održavanja Opus*Erp-a • Narodne novine d.d. - Vođenje razvoja i održavanja Opus*Erp-a • Stanić d.o.o. - Vođenje razvoja i održavanja Opus*Erp-a • HT d.d. - Vođenje razvoja i održavanja Opus*Erp-a • Zagrebačka banka - Vođenje razvoja i održavanja Opus*Erp-a

	2011. - 2014. IN2 d.o.o. Zagreb, Stariji Oracle razvojni inženjer Projekti: • Konzum d.d., Zagreb - implementacija i održavanje Oracle Retail sustava • Velpro d.d., Zagreb - implementacija i održavanje rješenja za veleprodaju • IN2 d.o.o., Zagreb - razvoj programskog okvira za razvoj aplikacija u Oracle ADF tehnologiji 2008. - 2011. IN2 d.o.o. Zagreb, Oracle eBS tehnički konzultant Projekti: • Hrvatska Lutrija d.d., Zagreb - održavanje Oracle eBS i IN2 aplikacija • Narodna banka Srbije - održavanje Oracle eBS-a, Financije • Dalekovod d.d., Zagreb - održavanje Oracle eBS-a • Plinacro d.o.o., Zagreb - održavanje Oracle eBS-a • Vodovod i kanalizacija d.o.o., Split - implementacija i održavanje Oracle eBS-a i IN2 aplikacija • Hrvatski zavod za mirovinsko osiguranje – održavanje Oracle eBS R12 (Financije) i upgrade postojećeg sustava Oracle eBS HR i IN2 Plaće sa 11.5.10 na R12 Tečajevi: • SQL/PLSQL, Designer and Developer training courses, IN2 d.o.o. , Croatia, 2008. • Oracle EBS - R12 Implement Oracle Workflow course, Oracle, Croatia, 2009. • J2SE, J2EE, ADF, JDeveloper 11g training courses, , IN2 d.o.o., Croatia, 2011. • Oracle RMS 13 Business Essentials: Foundation Data Ed1 course, Oracle, Croatia, 2011. • Oracle RMS 13 Technical Essentials Ed1 course, Oracle, Croatia, 2011. • Oracle RPM Business Essentials 13.1 Ed1 course, Oracle, Croatia, 2011. • Oracle Retail Integration Overview Ed1 course, Oracle, Croatia, 2011. • Oracle ReSA Essentials 13.1 Ed1 course, Oracle, Croatia, 2011. • Oracle ReIM Business Essentials 13.1 Ed1 course, Oracle, Croatia, 2011. • Oracle RPM TechnicalEssentials 13.1 Ed1 course, Oracle, Croatia, 2011.

	Certifikati: • Oracle Advanced PL/SQL Developer Certified Professional, Croatia, 2012. • Oracle Database 11g Administrator Certified Associate, Croatia, 2012. • Oracle Forms Developer Certified Professional, Croatia, 2012. • Oracle Certified Professional, Java SE 6 Programmer, Croatia, 2012. • Oracle Application Development Framework 11g PreSales Specialist, Croatia, 2016. • Oracle Application Development Framework 11g Sales Specialist, Croatia, 2016. Radovi: Bojić, A. Data integration model for ADF applications, Hroug - Oracle Conference, Rovinj, Croatia, 2014.
Popis radova:	• Bojić, A. Quantum Algorithm for Finding a Maximum Clique in an Undirected Graph, *Journal of Information and Organizational Sciences*, Vol. 36, No. 2, 2012. • Bojić, A. A New Quantum Game Based on CHSH Game, *Journal of Information and Organizational Sciences*, Vol. 37, No. 1, 2013. • Bojić, A. An approach to source code conversion of classical programming languages into source code of quantum programming languages, *Journal of Information and Organizational Sciences*, Vol. 38, No. 2, 2014.