Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tomaž Orač

Realizacija aritmetično-logičnih primitivov s strukturami kvantnih celičnih avtomatov

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

izr. prof. dr. Miha Mraz
MENTOR

doc. dr. Iztok Lebar Bajec
SOMENTOR

Ljubljana, 2007

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tomaž Orač

Realizacija aritmetično-logičnih primitivov s strukturami kvantnih celičnih avtomatov

POVZETEK

V diplomskem delu predstavljam svoje uvajanje v področje nanoračunalništva. Podano je le toliko teoretičnega ozadja, kolikor ga je nujno potrebnega za razumevanje osnovnih principov delovanja kvantnih celičnih avtomatov, brez nepotrebnega obremenjevanja s fizikalnimi podrobnostimi – le-te so predmet raziskav fizikov, ne računalničarjev. Poudarek je na analizi zmogljivosti orodja QCADesigner, ki se je na tem področju izkazalo za najbolj uporabno za preskušavanje novih idej. Preskusili smo ga na praktičnih primerih kvantnih celičnih avtomatov, natančneje, na realizaciji aritmetično-logičnih struktur, in ocenili njegovo delovanje s stališča zmožnosti simuliranja delovanja kvantnih celičnih avtomatov ter prijaznosti do uporabnika. Pri izbiri primerov struktur za realizacijo smo upoštevali njihovo razširjenost v sedanjih mikroprocesorjih. Verjamemo, da bodo enake strukture pomembne tudi v nanotehnološki izvedbi, saj so se do sedaj vedno znova dokazale kot osnovni gradnik računalniških sistemov.

Ključne besede: kvantni celični avtomati, QCADesigner, aritmetično-logične strukture

University of Ljubljana
Faculty of Computer and Information Science

Tomaž Orač

Implementation of arithmetic-logic primitives using quantum-dot cellular automata structures

ABSTRACT

In this thesis work I present my first steps into the field of nanocomputer science. Included is some theoretical background, but only what is absolutly neccesary for understanding the basic principles of quantum-dot cellular automata; quantum mechanical details are omitted – these should be the subject of research for physicists not computer engineers. The main focus of this thesis is the analysis of QCADesigner, which has proven to be very useful for testing new ideas. We have tested the tool on practical examples of quantum cellular automata, more precisely on the implementation of arithmetic-logic structures, and evaluated its capabilities of simulating quantum-dot cellular automata as well as its user friendliness. When choosing the examples of arithmetic-logic structures we have taken into consideration their presence in today's microprocessors. We believe that the same structures will play a significant role in nanocomputing, because they have proven themselves time and time again as the building blocks of computer systems.

Key words: quantum-dot cellular automata, QCADesigner, arithmetic-logic structures

ZAHVALA

Za strokovno vodstvo in nasvete bi se zahvalil mentorju izr. prof. dr. Mihi Mrazu in somentorju doc. dr. Iztoku Lebarju Bajcu. Zahvalil bi se tudi prijatelju Primožu Pečarju, ki mi je s svojo pomočjo precej olajšal izdelavo diplomskega dela.

Zahvalil bi se staršema za potrpljenje ter moralno in finančno podporo v času študija.

In nenazadnje, hvala tebi, Špela, za podporo in potrpljenje z risanjem diagramov po nareku, medtem ko sem sprehajal ledvične kamne po dnevni sobi.

— Tomaž Orač, Ljubljana, september 2007.

KAZALO

1 Uvod

Povezovanje izsledkov kvantne fizike z računalništvom obeta koristi, ki se slišijo že kar prelepo, da bi bile resnične. Vas živcirajo ogromne škatle, ki šumijo pod mizo? Jih preklinjate, ker so prepočasne, iz njih pa za nameček prav nepraktično štrli obilica kablovja? Sam lahko le dvakrat pritrdilno odgovorim. Pravzaprav je prva dobra lastnost, ki mi ta trenutek pade na pamet glede tega "čudežnega" strojčka ta, da jeseni nadomešča manjši grelec in prihranim pri centralni kurjavi.

Novo mejno področje med fiziko in računalništvom obljublja opazno zmanjšanje elektronskih naprav in grde škatle bi lahko zamenjale elegantnejše linije. Šum in ropot ventilatorjev bi utihnil in priljubljeno glasbo bi lahko poslušali tudi pri glasnosti, nižji od 50dB, pa se nežni *arpeggio* harfe še vedno ne bi izgubil v šumenju delovne sobe. Boljša polovica bi se sprva nekoliko pritoževala, da si ne more več hitro sušiti sveže nalakiranih nohtov v topli sapici z zadnje strani računalniškega ohišja, vendar bi se kmalu pomirila ob nižjem računu za elektriko. Končno bi lahko delali z računalnikom v "realnem času" – hitrost dela bi bila precej bolj odvisna od uporabnika, saj bi računalnik običajna opravila izvedel hitreje, kot bi ga mi utegnili preklinjati...

Že samo en izmed naštetih primerov bi me prepričal v koristnost nanotehnologije, ob vseh naštetih pa že iščem telefonsko številko podjetja, kjer bi tak izdelek lahko naročil. Pa je ni. Kako to? Ker je stvar šele pri (zelo dobri) ideji, praksa pa pri ugotavljanju katera od "nanoidej" se dejansko obnese. Nekje je pač treba začeti.

Pametni znanstevniki so kmalu ugotovili, da bi več ljudi dalo več idej in tako bi se področje hitreje razvijalo. Obstaja pa "manjši" problem, da kvantna fizika ni zanimiva oz. razumljiva vsakemu, ki ga to področje sicer zanima. Da bi navdušenim raziskovalcem pomagali prebiti led in olajšali delo, so naredili simulator, ki umazane fizikalne podrobnosti skrije pod uporabniški vmesnik, ki od daleč spominja na priljubljeno igro Tetris. Treba je le malo prestavljati in obračati kocke, pognati simulacijo in ugotoviti, ali smo imeli srečno roko. Če smo jo, bo mogoče nekoč naš umotvor nekomu dal idejo kako realizirati nekaj, kar bo uporabno tudi v praksi. Če pa iz te moke ne bo kruha, pa tudi prav. Prijateljem se še vedno lahko pohvalimo, da se ukvarjamo z nanotehnologijo in kvantno fiziko, ne da bi se pri tem preveč zlagali. Tako sem se tudi jaz javil za poskusnega zajčka in se po kratkem spoznavanju s kvantnimi celičnimi avtomati lotil odkrivanja radosti in pasti orodja QCADesigner.

V drugem poglavju tako podajam nekaj osnovne teorije kvantnih celic in kvantnih celičnih avtomatov, ravno toliko, kolikor jo je nujno potrebno za to, da se lahko lotimo postavljanja kvantnih struktur.

Tretje poglavje se ukvarja z orodjem QCADesigner, ki služi za simulacijo dogajanja v kvantnih celičnih avtomatih. Ker niti v angleškem, kaj šele v slovenskem jeziku, nisem našel dobrih navodil za uporabo, sem skušal čimbolj natančno in praktično opisati moje izkušnje z orodjem in tako prihraniti bodočim uporabnikom ponovno odkrivanje Amerike.

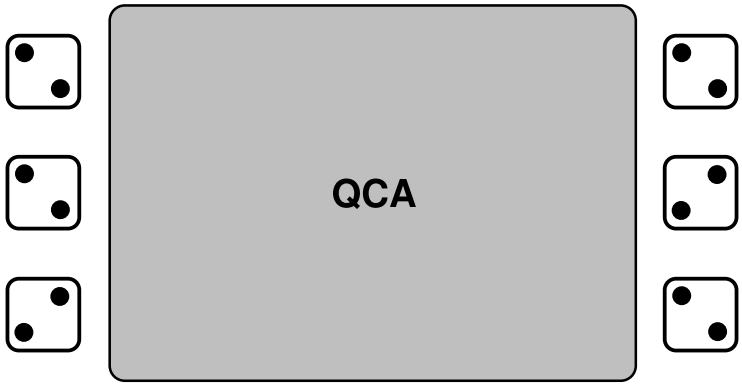
V zadnjem poglavju sem skušal uporabiti naučeno in predstaviti nekaj primerov uporabe simulatorja v praksi. Ker so me v času študija naučili, da je za hiter mikroprocesor potrebna hitra aritmetično-logična enota, sem se odločil, da s kvantnim celičnim avtomatom poskusim realizirati nekaj najpogosteje uporabljanih aritmetično-logičnih struktur.

2 Osnove kvantnih celičnih avtomatov

2.1 Kvantni celični avtomat

Kvantni celični avtomat (angl. *Quantum-dot cellular automaton*), s kratico QCA, deluje, kot je razvidno iz imena, po principih že znane arhitekture celičnih avtomatov. Bistveno se od njih razlikuje v tem, da ga sestavljajo kvantne naprave. Kvantni celični avtomat torej podaja način povezovanja kvantnih naprav v strukturo, ki bo sposobna opravljati določene uporabne funkcije [1]. Primer kvantnega celičnega avtomata predstavlja slika 2.1.

V primeru na sliki se vhodi (vhodne celice) nahajajo na enem robu polja kvantnega celičnega avtomata, izhodi (izhodne celice) pa na drugem. Možen je tudi primer, ko obstaja več vhodnih in izhodnih robov, vhodi in izhodi pa so lahko tudi skupaj na istem robu. Postavljanje vhodnih črk avtomata poteka tako, da se s pomočjo zunanjih vplivov določi stanje vhodnih celic. Stanje se nato tekom procesiranja izhodne vrednosti ne spreminja (je *fiksirano*). Ker se le-te nahajajo na robu polja, se ta način procesiranja imenuje *robno gnano procesiranje* (angl. *edge-driven computation*). Izhodne celice pa, nasprotno vhodnim, niso fiksirane, ampak se zaradi medsebojnih vplivov celic kvantnega



Slika 2.1 Primer kvantnega celičnega avtomata. Celice na levem robu predstavljajo vhode (robne pogoje), celice na desnem pa izhode, od koder se prebere rezultat procesiranja. "Notranjost" polja pa sestavlja kvantna celična struktura, ki ustreza reševanju danega problema.

celičnega avtomata postavijo v določeno stanje (*polarizacijo*) in so namenjene branju rezultatov procesiranja. Procesiranje tako poteka po naslednjih korakih:

1. Postavljanje vhodnih črk avtomata s fiksiranjem stanja vhodnih celic.
2. Čakanje, da kvantni celični avtomat preide v *osnovno stanje*.
3. Branje rezultatov procesiranja z zaznavanjem stanja izhodnih celic.

Iz povedanega sledi, da kvantni celični avtomat deluje (prehaja med stanji) po principu *procesiranja z osnovnim stanjem* (angl. *computing with the ground state*). Osnovno stanje avtomata je stanje, v katerem po določenem času avtomat, pri danih (fiksnih) vhodih, obmiruje – vse celice avtomata so v takih stanjih, da zagotavljajo minimalno energijsko stanje avtomata. Sprememba vrednosti vhodov (*robnih pogojev*) povzroči, da kvantni celični avtomat ni več v osnovnem stanju, ampak v *vzbujenem stanju*. Iz takega stanja nato preide z umirjanjem preko nižjih vzbujenih stanj v novo osnovno stanje, ki ustreza danim robnim pogojem in je energijsko najbolj ugodno. Umirjanje torej poteka preko fizikalnih procesov, ki prerazporedijo elektrone v celicah kvantnega celičnega avtomata. Iz povedanega sledi, da mora imeti kvantni celični avtomat tako strukturo, da njegovo osnovno stanje ustreza rešitvi problema, danega z vhodnimi podatki.

Uporaba kombinacije robno gnanega procesiranja in procesiranja z osnovnim stanjem ima eno veliko prednost. Vso potrebno informacijo za reševanje nekega problema je potrebno pripeljati le na robove polja od koder se rezultati tudi berejo. Torej povezave

v "notranjost" kvantnega celičnega avtomata niso potrebne, kar je posebej pomembno, saj so kvantne naprave zelo majhne. Če bi namreč bilo potrebno narediti povezave med posameznimi celicami znotraj avtomata, bi to predstavljalo velik problem.

Ta princip je ravno nasproten tistemu, ki se največ uporablja v konvencionalnem procesiranju, ki je običajno realizirano z visoko vzbujenimi neravnovesnimi stanji. Na katerikoli točki se namreč lahko v sistem dovaja energija, saj so na voljo povezave do posameznih elementov (naprav) v sistemu. Prednost takega načina je večja neobčutljivost na zunanjo temperaturo, zato lahko taki sistemi delujejo na sobni temperaturi. Po drugi strani pa ravno to povzroča veliko težav pri njihovem pomanjševanju do nanometrskih razsežnosti.

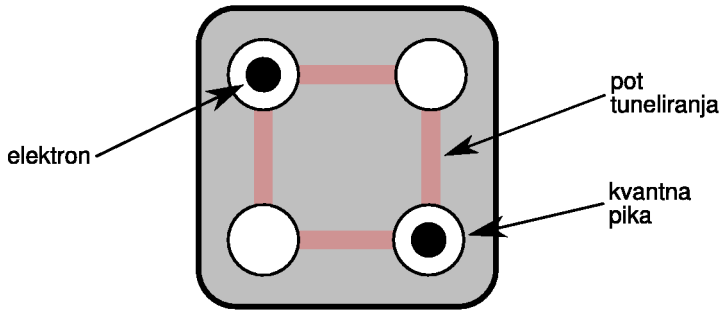
Težav s pomanjševanjem kvantni celični avtomat nima, je pa zelo občutljiv na temperaturne vplive. Problem je energijska razlika med osnovnim in prvim vzbujenim stanjem. Če je ta premajhna, se lahko zgodi, da avtomat ne doseže osnovnega stanja, ampak ostane ujet v enem od vzbujenih stanj. To pa seveda ne ustreza iskani rešitvi problema. Zaradi omenjenega kvantni celični avtomati zaenkrat delujejo le pri zelo nizkih temperaturah (nekaj stopinj Kelvina). Natančneje, za velikost kvantne pike, ki jo je trenutna tehnologija sposobna izdelati, velja, da deluje pri temperaturi tekočega helija [1].

2.2 Kvantna celica

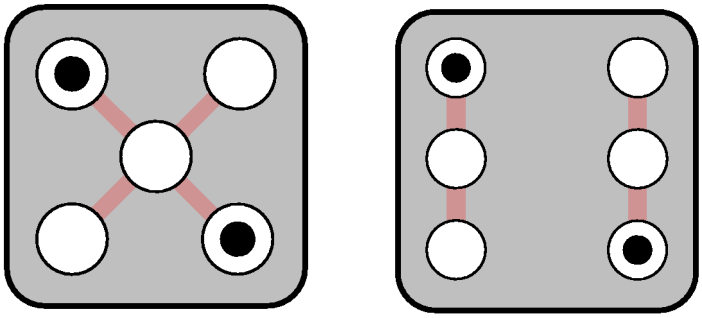
Osnovni gradnik kvantnega celičnega avtomata je *kvantna* ali *QCA celica* [2]. Višjenivojski diagram take celice prikazuje slika 2.2. V splošnem sestavlja celico množica *kvantnih pik*. To so območja v kvantni celici, ki imajo potencialno energijo večjo kot je termična energija okolice, in lahko zaradi tega zadržijo posamezne naboje (elektrone).

Tipično se uporablja najenostavnejša oblika celice, to je celica s štirimi kvantnimi pikami, ki so postavljene v vogale kvadrata. Obstajajo tudi kompleksnejše oblike, kot sta npr. celica s petimi oz. šestimi kvantnimi pikami (slika 2.3). Ker za realizacijo dvonivojske logike popolnoma zadostuje najenostavnejša oblika celice, se le-ta tudi najpogosteje uporablja.

V kvantni celici se nahajata natanko dva *elektrona*, ki lahko prehajata med različnimi kvantnimi pikami po principu *tuneliranja elektronov*. Poti, po katerih lahko elektrona tunelirata, so na diagramu predstavljene s povezavami med posameznimi kvantnimi pikami. Tuneliranje elektronov je popolnoma nadzorovano z višanjem in nižanjem potencialnih



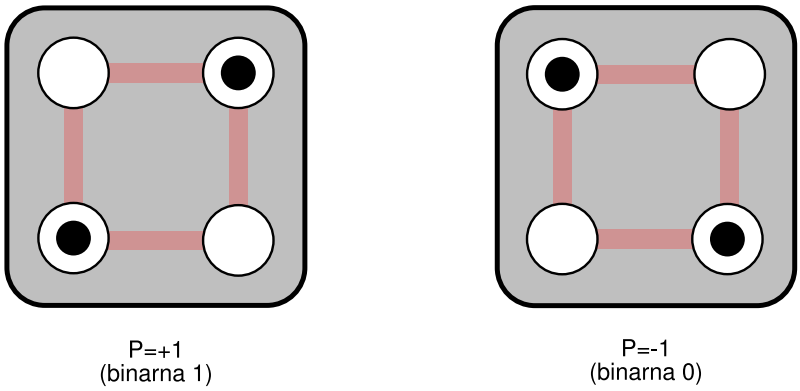
Slika 2.2 Višjenivojski diagram kvantne celice. Celico sestavljajo štiri kvantne pike, ki se nahajajo v vogalih kvadrata. Pike povezujejo štiri poti tuneliranja, po katerih lahko tunelirata dva elektrona.



Slika 2.3 Kvantna celica s petimi oz. šestimi kvantnimi pikami.

pregrad med kvantnimi pikami. Pregrade morajo biti tako visoke, da se lahko naboj (elektron) premika le s tuneliranjem in se zato vedno nahaja le v kvantnih pikah in ne na poteh, preko katerih poteka tuneliranje. Zaradi odbojne Coulombove sile se skušata elektrona medseboj čimbolj oddaljiti. Kadar ni zunanjih vplivov, se zato elektrona nahajata v nasproti ležečih vogalih celice (iz meja celice ne moreta pobegniti), saj sta tam v minimalnem energijskem stanju. Ker ima celica štiri kvantne pike (vogale), sta tako vsak trenutek možni dve osnovni razporeditvi elektronov (stanji celice ali polarizaciji), ki ju prikazuje slika 2.4.

Omenjeni polarizaciji sta idealni za predstavitev logične enke in logične ničle, kar je osnova za binarno predstavitev informacije. Po dogovoru ustreza polarizacija $P = +1$ logični enki in polarizacija $P = -1$ logični ničli. Obe polarizaciji imata, kadar ni zunanjih vplivov, enako elektrostatično energijo. Katera od obeh je v kvantnem celičnem avtomatu



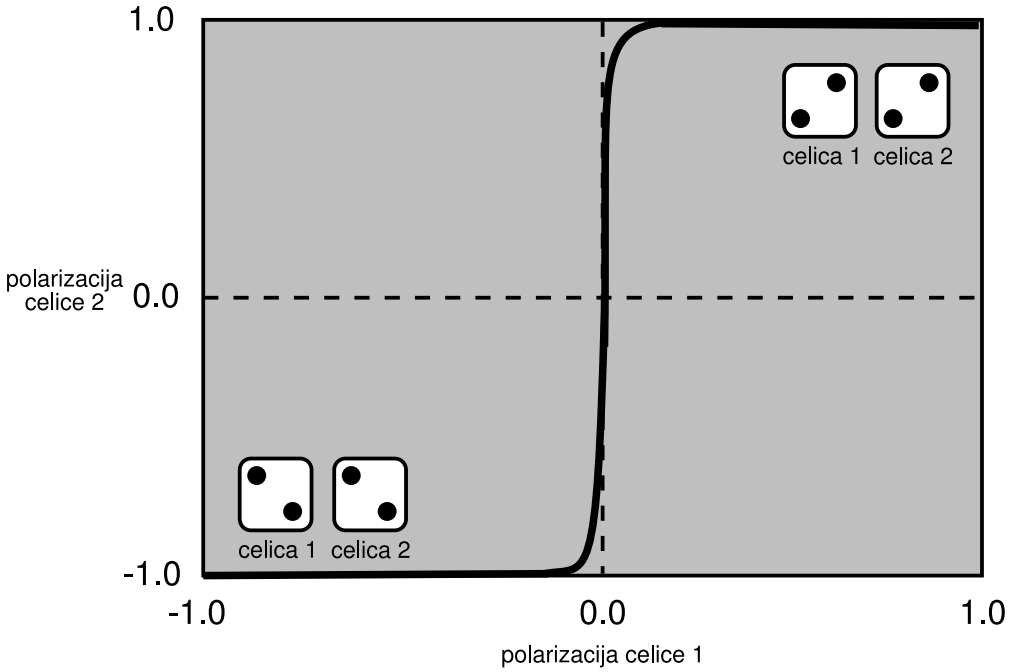
Slika 2.4 Možni polarizaciji v kvantni celici s štirimi kvantnimi pikami. Elektrona se skušata zaradi Coulombove odbojne sile medseboj čimbolj oddaljiti in se zato nahajata v vogalih celice.

v nekem trenutku tista z nižjo energijo in tako bolj verjetna, je zato odvisno le od vplivov sosednjih celic. Na ta način je omogočen prenos informacije med celicami in s tem procesiranje. Če so pregrade, ki kontrolirajo tuneliranje, dovolj visoke, je interakcija med sosednjimi celicami (*prenosna funkcija*) zelo nelinearna (glej sliko 2.5). To pomeni, da polarizacija neke celice zelo vpliva na polarizacijo sosednje celice – že majhna polarizacija izbrane celice povzroči skoraj popolno polarizacijo njene sosede. Nelinearna prenosna funkcija ima obenem še to dobro lastnost, da zagotavlja da bo spreminjanje stanj celic potekalo zelo hitro.

2.3 Osnovne kvantne celične strukture

Praktična uporabnost koncepta kvantnih celičnih avtomatov se najenostavneje pokaže tako, da se dokaže, da se z množico kvantnih celic, ki sestavljajo *kvantno celično strukturo*, lahko realizira funkcijsko poln sistem Boolove algebre. Znano je namreč, da se s funkcijsko polnim sistemom lahko realizira poljubno logično funkcijo, kar pomeni, da lahko sistem rešuje splošne probleme.

Funkcijsko polnih sistemov Boolove algebre je več. V CMOS tehnologiji se v računalniških strukturah in sistemih najpogosteje uporabljata Shefferjev ali Piercov funkcijsko polni sistem, torej NAND ali NOR logična vrata. Na področju kvantnih naprav pa se največ uporablja osnovni nabor, ki izhaja iz Boolove algebre in ga sestavljajo logične funkcije negacije, konjunkcije in disjunkcije. Slednji sta realizirani kot tako imenovana *majoritetna vrata* (angl. *majority gate*), zato ne preseneča da le-ta predstavljajo najo-

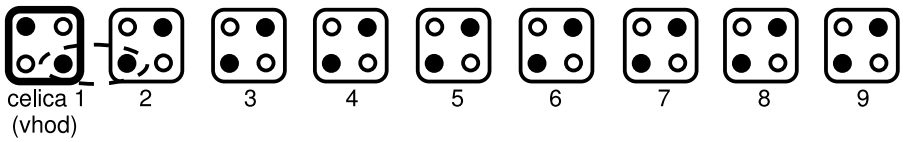


Slika 2.5 Odziv izbrane celice na spremembo polarizacije sosednje celice.

snovnejša logična vrata v kvantnih celičnih strukturah.

2.3.1 Kvantna linija

Najpreprostejša, a hkrati ena izmed najpomembnejših struktur, je linija oziroma žica (angl. *wire*). Sestavlja jo zaporedje kvantnih celic, po katerih, v primeru na sliki 2.6, potuje signal od leve proti desni. Signal se po žici širi zaradi medsebojnih interakcij med celicami. Obstajata 45-stopinjska in 90-stopinjska žica, razlikujeta pa se po kotu rotacije celic, ki ju sestavljajo.

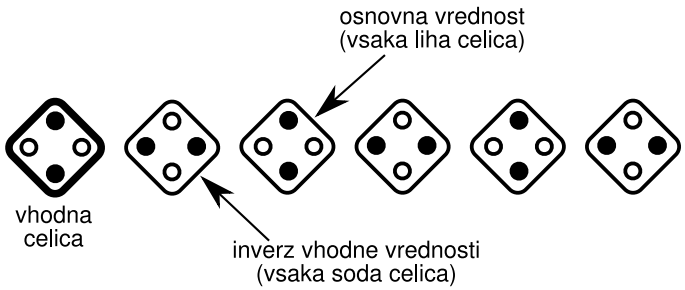


Slika 2.6 Najenostavnejša kvantna struktura – 90-stopinjska žica.

Celica 1 pri 90-stopinjski žici predstavlja vhodno celico in ima v primeru s slike 2.6 fiksno polarizacijo $P = -1$, celica 2 pa ima v tem primeru polarizacijo $P = +1$. Polarizacije celic 2-9 niso fiksne. Zaradi odboja med elektroni v celicah 1 in 2, celica 2 spremeni

polarizacijo. Posledično to povzroči odboj med elektroni v celicah 2 in 3 in spremembo polarizacije celice 3. Enak proces se nadaljuje po celotni dolžini žice. Osnovno (energijsko minimalno) stanje nastopi takrat, ko je Coulombov odboj med elektroni najmanjši možen. V obravnavanem primeru je to takrat, ko vse celice uskladijo svojo polarizacijo z vhodno celico in tako prenesejo informacijo o vhodni polarizaciji preko žice do izhodne celice (celice 9).

Pri 45-stopinjski žici je dogajanje podobno, le da polarizacija celic vzdolž žice alternira med $P = +1$ in $P = -1$, saj takšna razporeditev zagotavlja osnovno (energijsko minimalno) stanje. Dobra stran take žice je ta, da sta na njej na voljo tako osnovna, kot tudi inverzna logična vrednost (glej sliko 2.7) in tako ni potrebe po dodatnih negatorjih.

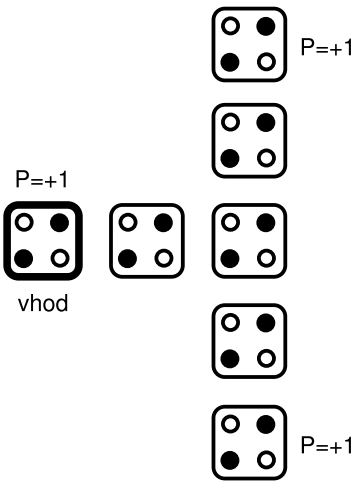


Slika 2.7 45-stopinjska žica in način odjema osnovne in inverzne logične vrednosti.

S stališča praktične uporabe žice je pomembno tudi dejstvo, da žica dovoljuje določeno odstopanje od popolne poravnanosti celic v nizu, pa kljub temu brez napak prenaša informacijo. Kolikšno je to odstopanje je odvisno predvsem od tehnologije, s katero je kvantna celica implementirana. Podobno je od tehnologije odvisna tudi največja možna dolžina žice. Ocenjujejo pa, da bodo pri temperaturah, ki so blizu sobni, možna polja s številom celic velikostnega reda 10^5 [2].

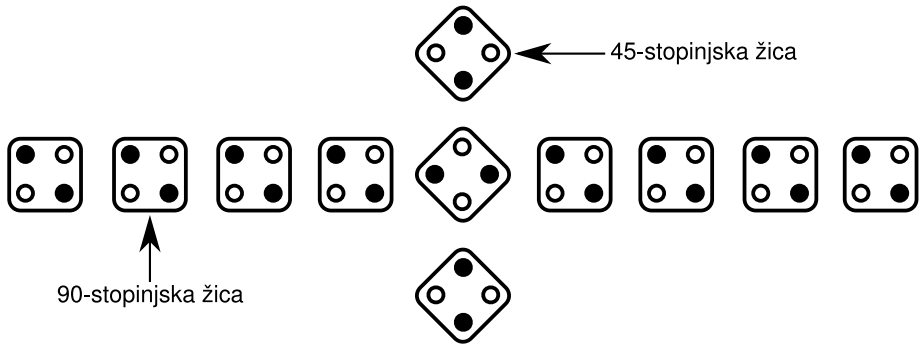
Žice niso omejene samo na enodimenzionalen niz celic. Brez težav se lahko lomijo pravokotno, postavljajo v diagonale, itd. Enako enostavno je tudi razmnoževanje signala (angl. *fan-out*), ki ga npr. potrebujemo na večih koncih strukture. V današnjih klasičnih računalniških strukturah je to rešeno z ojačevalniki. Ti poskrbijo, da je na voljo dovolj električnega toka (energije) za vse kopije signala in tako preprečijo preobremenitev izhoda. Pri kvantnih strukturah tega problema ni, saj se signal (informacija) med celicami prenaša breztokovno, med drugim pa se zaradi izrazite nelinearne prenosne funkcije tudi ojačuje. Razmnoževanje siganala je zato realizirano z enostavno razcepitvijo žice (glej

sliko 2.8) in ni nobene potrebe dovajanju dodatne energije.



Slika 2.8 Razmnoževanje signala z enostavno razcepitvijo žice.

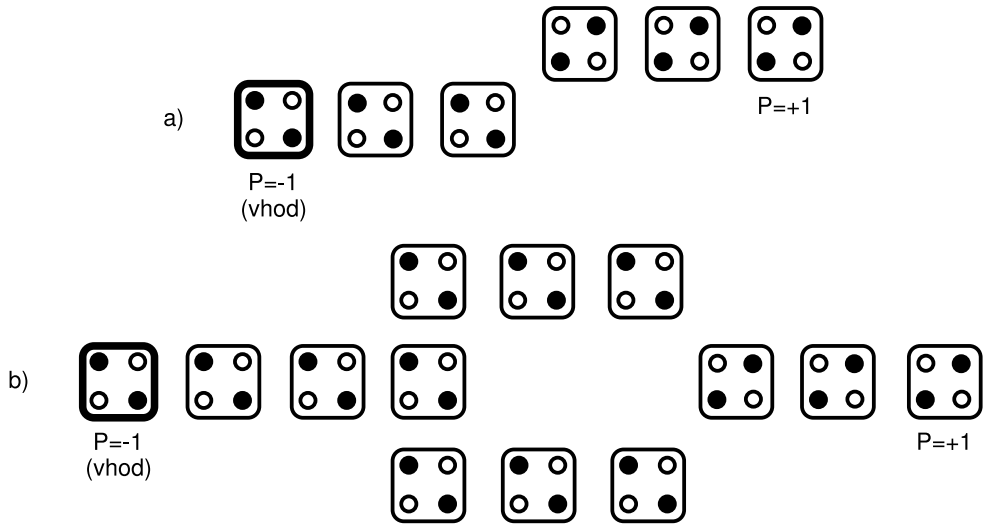
Pri obravnavi kvantnih žic je potrebno omeniti še eno pomembno lastnost. Dve žici se lahko križata v isti ravnini, ne da bi se signala, ki potujeta po obeh žicah, medseboj motila (slika 2.9). Edina omejitev je, da morata biti žici različnih tipov – križata se lahko le 45- in 90-stopinjska žica, ne pa tudi dve 90-stopinjski. Ta lastnost je edinstvena kvantnim žicam. V klasičnih računalniških strukturah to ni mogoče in križanje linij zahteva uporabo večplastnih tiskanih vezij.



Slika 2.9 Križanje dveh žic v isti ravnini.

2.3.2 Negator

Poleg majoritetnih vrat in linij je zelo uporabljana struktura negator (angl. *inverter*). Izkorišča lastnost 45-stopinjske žice, to je, da se celici, ki se "stikata" z vogaloma, postavita v nasprotni polarizaciji. Možnih je več različnih implementacij negatorja, razlikujejo pa se po občutljivosti na motnje morebitnih drugih celic v bližini. Primer na sliki 2.10a se uporablja predvsem pri kvantnih žicah, kjer v žico vstopa nek signal, na koncu žice pa je potrebna negacija tega signala. Izvedba prikazana na sliki 2.10b pa je zaradi robustnosti (manjše občutljivosti na motnje iz okolice) primernejša za uporabo znotraj večjih struktur.

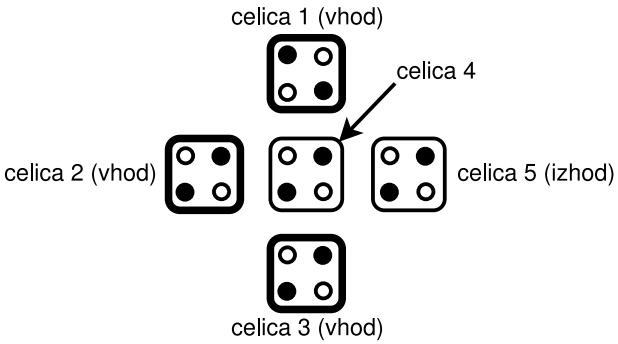


Slika 2.10 Dve izmed možnih realizacij negatorja. Realizacija s slike a) je značilna predvsem za QCA žice, medtem ko se realizacija s slike b) več uporablja v večjih QCA strukturah.

2.3.3 Majoritetna vrata

Strukturo z imenom *majoritetna* ali *večinska* vrata prikazuje slika 2.11. Princip njihovega delovanja je zelo enostaven. Izhod vrat podaja večinsko stanje vhodov – če je večina vhodov na logični enki, je tak tudi izhod. Če pa je število vhodov na logični ničli večje od števila vhodov na logični enki, je na logični ničli tudi izhod. S kvantnimi celicami je to realizirano tako, da predstavljajo celice 1, 2 in 3 na sliki 2.11 vhode, celica 5 izhod, celica 4 pa je potrebna za pravilno delovanje strukture in ni neposredno dosegljiva.

Kot je bilo povedano pri opisu kvantnega celičnega avtomata, predstavlja rešitev



Slika 2.11 Najosnovnejša logična vrata v kvantnih strukturah – majoritetna vrata.

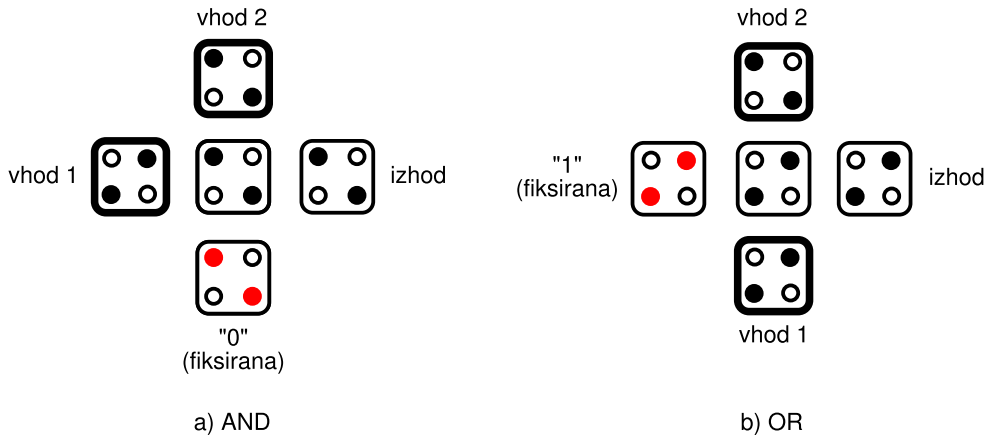
danega problema osnovno stanje strukture. V primeru majoritetnih vrat je to takrat, ko sta celici 4 in 5 v najnižjem energijskem stanju. Celica 4 ga doseže takrat, ko svoje stanje uskladi s stanjem večine vhodnih celic, celica 5 pa takrat, ko svoje stanje uskladi s stanjem celice 4. V tem stanju je namreč odbojnost med elektroni vhodnih celic in elektroni celic 4 in 5 najmanjša. S tem delovanje strukture ustreza definiciji majoritetnih vrat.

2.3.4 Logična vrata AND in OR

Naslednji par pogosto uporabljanih struktur so logična vrata AND in logična vrata OR. Tudi tu je, kot pri negatorju, možnih več različnih realizacij, najenostavneša je kar z uporabo majoritetnih vrat. Funkcija AND je implementirana tako, da je eden od vhodov v majoritetna vrata (celice 1, 2 ali 3 na sliki 2.11) fiksiran na logično 0 (slika 2.12a). Podobno je funkcija OR realizirana tako, da je eden od vhodov fiksiran na logično 1 (slika 2.12b).

S tem, ko so na voljo negator ter logični funkciji AND in OR, pa je tudi dokazano, da je mogoče s kvantnimi celicami implementirati funkcijsko poln sistem in tako realizirati poljubno logično funkcijo.

Potencial koncepta kvantnih celičnih avtomatov sega precej dlje, kot je le realizacija osnovnih Boolovih operacij. Obstaja kar nekaj kvantnih struktur, ki olajšajo realizacijo logičnih funkcij in zmanjšajo prostorsko zahtevnost. Nekaj primerov takih struktur opisujejo naslednji razdelki.



Slika 2.12 Implementacija logičnih funkcij AND in OR z kvantnimi celicami.

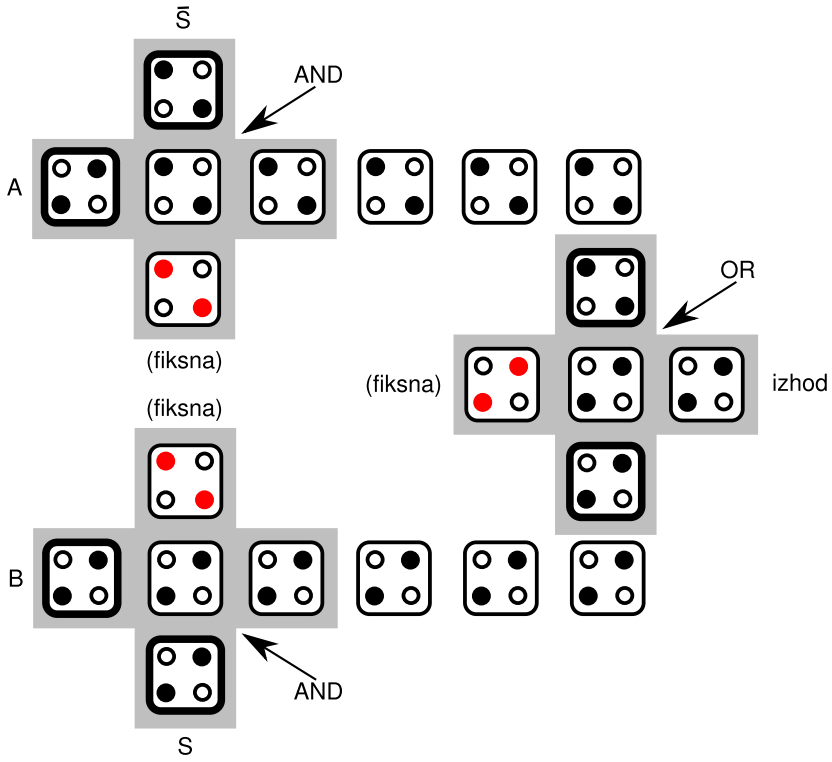
2.3.5 Primer kompleksnejšega vezja

Kot primer uporabe predstavljenih kvantnih celičnih struktur je podan dvovhodni multiplekser, ki ga sestavljajo ena logična vrata OR in dvoje logičnih vrat AND (slika 2.13). Z A in B sta označena vhoda v multiplekser, signal S pa izbira med vhodoma. Ker sta logični funkciji AND in OR implementirani z majoritetnimi vrati (siva področja na sliki), so tri celice fiksirane na logično 0 oz. 1.

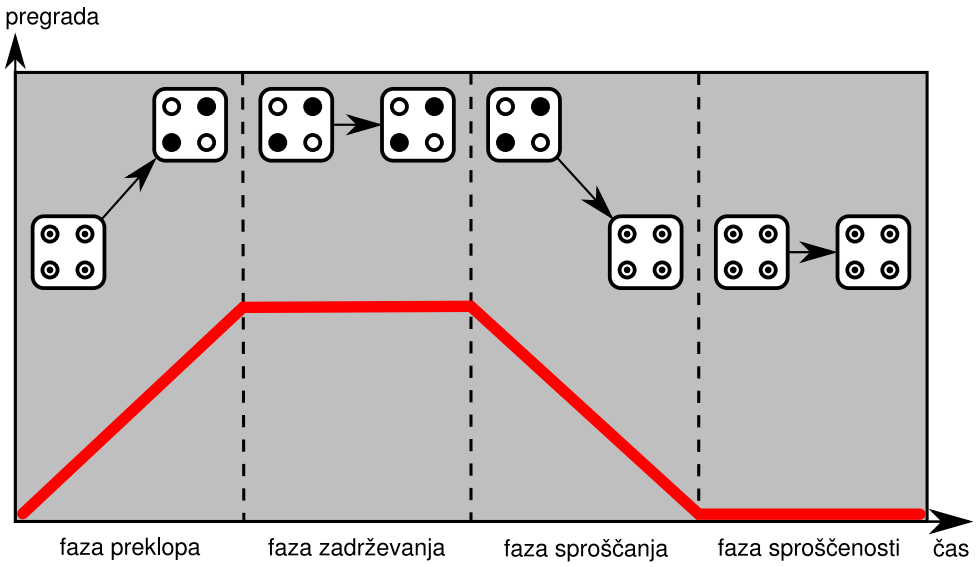
2.4 Urin signal v kvantnih celičnih strukturah

Urin signal ima v kvantnih celičnih strukturah precej drugačno vlogo kot v klasičnih strukturah, ki temeljijo na CMOS tehnologiji. V CMOS tehnologiji je to predvsem poseben signal, ki natančno kontrolira časovni potek dogajanja v strukturi. Tako npr. kontrolira kdaj se biti prenesejo v ali iz pomnilniških elementov (angl. *flip-flops*). Pri tej vlogi zadostujeta dve fazi oz. stanji urinega signala (visoko in nizko). Eno od teh stanj je navadno aktivno, drugo pa ne. V aktivnem stanju lahko npr. poteka vpisovanje podatkovnega bita v pomnilniško celico, v neaktivnem pa to ni mogoče.

V strukturah, ki so osnovane na kvantni tehnologiji, pa služi ura za močnostno ojačenje (angl. *power gain*) signala [3]. Tako kot vsak realen sistem tudi kvantne strukture niso energijsko idealne in imajo zato določene izgube. V daljših nizih kvantnih celic neizogibno pride do izgub energije signala. To izgubljeno energijo je potrebno na nek način nadomestiti iz nekega vira iz okolice kvantnih celic ter tako zagotoviti, da se



Slika 2.13 Implementacija dvovhodnega multiplekserja z uporabo logičnih funkcij AND in OR.



Slika 2.14 Štiri faze ure v kvantnih celičnih avtomatih.

kodirana informacija pri prenosu med celicami ne popači. To je naloga urinega signala.

Tudi način vstopanja urinega signala v kvantno celično strukturo je drugačen kot pri CMOS strukturah. Ne gre za klasično linijo, ki bi bila napeljana do (skoraj) vsakega elementa vezja, ampak gre tipično za *električno polje*. To kontrolira pregrade znotraj kvantnih celic kar posredno kontrolira premike elektronov med posameznimi kvantnimi pikami znotraj celice.

Za uspešno opravljanje dane funkcije, ima v kvantnih celičnih strukturah ura *štiri faze* [4] (slika 2.14):

1. faza preklopa (angl. *switch*),
2. faza zadrževanja (angl. *hold*),
3. faza sproščanja (angl. *release*) ter
4. faza sproščenosti (angl. *relax*).

Te faze so posledica dogajanja v celici oz. spreminjanja višine pregrad. Te so lahko dvignjene (faza 2) ali spuščene (faza 4), lahko pa so v fazi dvigovanja (faza 1) ali spuščanja (faza 3), kar da štiri faze QCA ure.

V prvi fazi, *fazi preklopa*, se izvrši dejansko preklapljanje oz. vzpostavitev stanja celice. Sprva so pregrade med kvantnimi pikami nizke in celica je nepolarizirana. Nato

se pregrade pričnejo dvigovati in celica postaja polarizirana v skladu s stanjem gonilnikov (vhodnih celic). Do konca te faze so pregrade že tako visoke, da onemogočijo kakršno koli tuneliranje elektronov in stanje celic je tako fiksirano.

V naslednji fazi, *fazi zadrževanja*, se višina pregrad vzdržuje in celica ostaja v stanju iz predhodne faze. Tako se lahko stanja celic, ki so se vzpostavila v prvi fazi, uporabijo kot gonilniki (vhodi) drugih delov strukture (vezja).

Tretja faza, *faza sproščanja*, ponovno prične z zniževanjem pregrad in celice se lahko sprostijo v nepolarizirano stanje.

V zadnji fazi, *fazi sproščenosti*, nato ostanejo pregrade nizke in celica je v nepolariziranem stanju.

Ena *urina perioda* torej traja toliko časa, da gre celica skozi vse štiri faze.

V praksi bi bilo zelo težko implementirati strukturo, kjer bi imela vsaka celica svojo uro, saj se pojavijo problemi zaradi velikega števila povezav med posamezno celico in pripadajočo uro. Vendar to ni posebej huda omejitev, saj se izkaže, da v praksi ni potrebe po tem. Zadostuje poenostavitev, ki strukturo razdeli na posamezna *urina področja*. Zanje velja naslednje:

1. Vse celice v istem področju si delijo isto uro, se pravi da so vse hkrati v isti urini fazi (en potencial kontrolira višino pregrad vseh celic v področju).

2. Sosednji področji sta v nekem trenutku v sosednjih urinih fazah.

Iz povedanega sledi, da opravlja ura v kvantnih celičnih avtomatih dve nalogi:

1. Področja, ki so v različnih fazah, omogočajo, da del strukture v eni fazi izračuna neko logično funkcijo (faza preklopa) in v naslednji shrani rezultate (fiksira stanja celic z dvigom pregrad – faza zadrževanja). Tako so ti rezultati na voljo na izhodih tega področja in jih lahko ostala področja uporabijo kot vhode za nadaljno procesiranje. Medtem ko so celice nekega področja v fazi računanja logične funkcije, ura poskrbi še za to, da so celice sosednjih področij v nepolariziranem stanju (fazi sproščenosti). To zagotavlja, da ne pride do neželenih vplivov sosednjih celic na celice, ki trenutno izračunavajo logično funkcijo, obenem pa že nudi preprosto cevovodno delovanje strukture (angl. *pipelining*).

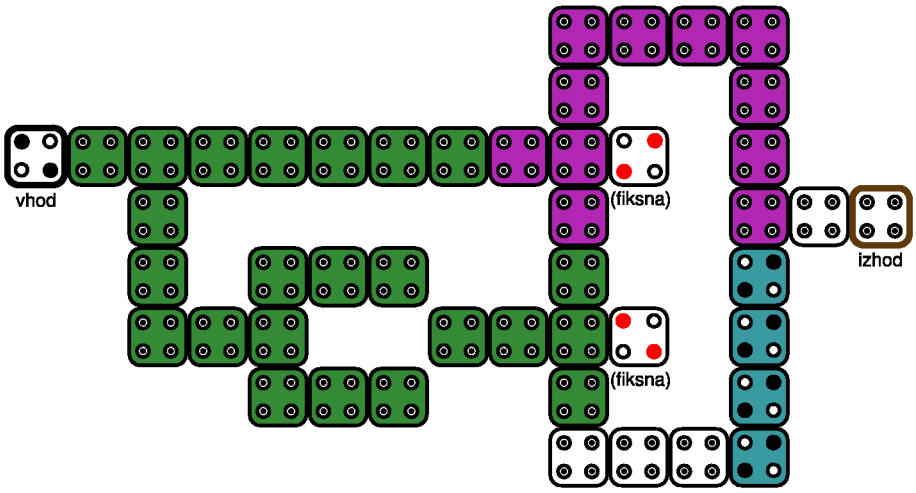
2. Pomembno je tudi ojačenje (angl. *gain*), ki nastane kadar ura drži celice v določeni polarizaciji. To je analogno močnostnemu ojačenju (angl. *power gain*), kot je po-

znano v klasičnih elektronskih napravah, in restavraciji signalov s pomočjo pull-up oz. pull-down uporov [3].

2.4.1 Pomnilna celica

Z do sedaj opisanimi strukturami, ki ne izkoriščajo ure, je mogoče implementirati le kombinatorične logične strukture. S kvantnimi strukturami pa je mogoče doseči tudi pomnjenje informacije, kar je osnova za sekvenčne logične strukture.

V nasprotju s prvotnimi predvidevanji [1] ena sama kvantna celica oz. polje celic *ne* deluje kot pomnilna celica. Da celica ostane v neki polarizaciji in na ta način hrani informacijo, je namreč potrebno nenehno osveževanje trenutnega stanja. To je mogoče doseči z večimi celicami, povezanimi v zanko – podobno kot pri konvencionalni CMOS-tehnologiji. Primer take pomnilne celice prikazuje slika 2.15.



Slika 2.15 Pomnilna celica. Različna urina področja so označena z barvami: zelena predstavlja prvo področje (faza preklopa), viola drugo (faza zadrževanja), bela tretje (faza sproščanja) in modra četrto (faza sproščenosti). Pri modrih celicah je vidna še ena posebnost–začetno stanje celice je preddefinirano (ni nevtralno).

Obstajajo tudi drugačne implementacije pomnilnih celic, ki izkoriščajo posebne lastnosti kvantnih celičnih struktur. Primer take D-celice je podrobneje predstavljen v tretjem poglavju. Vsem implementacijam pa je skupno to, da za delovanje potrebujejo urin signal.

3 Orodje QCADesigner

3.1 Uvod

QCADesigner[1] je orodje za postavitev in simulacijo delovanja kvantnih celularnih avtomatov. Razvito je bilo v laboratoriju ATIPS na univerzi v Calgaryju, z namenom, da pride vedno večja množica raziskovalcev s področja nanotehnologije do orodja, ki bo omogočalo enotnejši in hitrejši razvoj. Zaradi tega ima orodje grafični uporabniški vmesnik, hkrati pa dovolj zmogljive algoritme simulacije, da so rezultati uporabni v realnosti. V začetku je bilo orodje namenjeno predvsem skupnemu raziskovanju kvantnih celičnih struktur na univerzi v Calgaryju in Notre-Damu, Centru za nanoznanost in tehnologijo ter Tehnični univerzi v Budimpešti. Kmalu pa je postalo širše uporabljano, saj so ga objavili na Internetu kot prosto dostopno odprtokodno programsko opremo. Tako si lahko orodje vsak uporabnik naloži na svoj računalnik, prispeva k nadaljnemu razvoju oz. ga prilagaja svojim potrebam. Za izrisovanje grafičnega vmesnika se uporablja odprtokodna knjižnica GTK+[2] (angl. *The GIMP toolkit*).

[1] http://www.qcadesigner.ca
[2] http://www.gtk.org

Razvoj se je najprej začel v okoljih Unix (ter Linux), kasneje pa so razvijalci pričeli vzporedno izdajati tudi različice za okolje Microsoft Windows. Ker je orodje še vedno v razvojni fazi, delovanje seveda ni povsem brez napak, pa tudi nekatere funkcije niso še povsem dodelane. Potrebno je opozoriti, da ima različica za okolja Unix (Linux) nekaj manj pomanjkljivosti in je pri simulacijah po moji oceni za približno tretjino hitrejša. To gre najbrž pripisati dejstvu, da se razvija in uporablja dlje kot različica za Microsoft Windows.

Pri svojem delu sem uporabljal najnovejšo različico, ki je bila v času pisanja na voljo, to je verzija 2.0.3 v okolju Linux.

3.2 Uporabniški vmesnik

Uporabniški vmesnik je dokaj standarden, tako da je po priloženi pomoči potrebno poseči le pri naprednejših funkcijah. Zaradi uporabe knjižnice GTK+ je tudi zelo hiter in odziven. Knjižnica GTK+ je v okoljih Unix (Linux) že prednaložena, za okolje Microsoft Windows pa jo je potrebno naložiti ločeno od osnovnega programa.

Ko zaženemo program, se odpre glavno okno podobno tistemu na sliki 3.1. Slika prikazuje QCADesigner verzije 2.0.3 v okolju Ubuntu Linux 6.06 LTS[3] z naloženim primerom enostavne kvantne celične strukture. Izgled se od različice do različice orodja nekoliko razlikuje, vendar ne toliko da bi bilo to moteče.

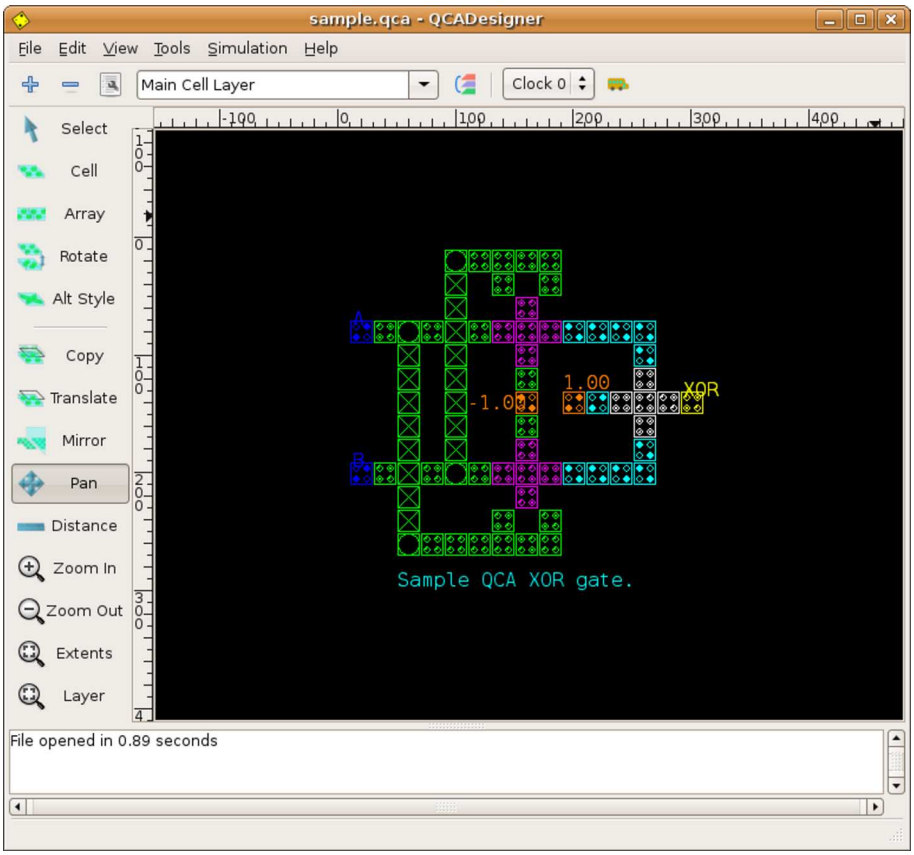
Kot je razvidno iz slike, je izgled orodja dokaj klasičen – vrstica z glavnimi meniji, vodoravna in navpična orodna vrstica ter delovno področje, v katerem postavljamo kvantno celično strukturo. Pri dnu zaslona je še informacijsko okno, kjer se sproti izpisujejo informacije o trenutnem dogajanju v programu in razni podatki o strukturi.

Pri delu z orodjem sta najbolj uporabni vrstica z glavnimi meniji ter navpična orodna vrstica. Nekatere opcije v glavnih menijih imajo enake funkcije kot je v navadi pri drugih programih in jih ni potrebno posebej opisovati. Druge pa so posebnost QCADesignerja in so zato v nadaljevanju podrobneje razložene.

3.2.1 Meni File

V meniju File je poleg običajnih izbir za odpiranje in shranjevanje dokumentov nekoliko posebna opcija *Print*. Ta poleg tiskanja na tiskalnik omogoča tudi "tiskanje" (izvoz) v datoteko formata *PostScript* oz. *Encapsulated PostScript*, kar je zelo uporabno za npr.

[3] http://www.ubuntu.org



Slika 3.1 Glavno okno QCADesignerja.

predstavitve narejenih struktur ne da bi potrebovali prednaložen osnovni program. Pri izvažanju strukture pa je potrebno nekoliko eksperimentirati z nastavitvami preslikave nanometrske skale, ki jo uporablja program, v centimetersko skalo, ki jo pričakuje tiskalnik. Navadno je struktura sprva prevelika ali premajhna za čitljiv prikaz na listu papirja formata A4. Dodatno otežuje delo dejstvo, da opcija *Print Preview* (še) ne deluje in so rezultati spreminjanja nastavitev vidni šele, ko je končna datoteka prikazana v kakšnem drugem grafičnem programu, zaradi česar izvažanje zahteva malo več časa in potrpljenja.

Ena od večjih sprememb v novejših različicah orodja (npr. različice od 2.0.* dalje) je ta, da orodje ne podpira več dela s projekti. V starejših različicah je vsaka postavljena struktura pripadala nekemu projektu. Ker pa v projekt ni bilo mogoče imeti vključenih več datotek s strukturami hkrati, je bila ta opcija odvečno obremenjevanje uporabnika, saj ni bilo od takega pristopa nobene koristi. Novejše različice so zato poenostavljene tako, da postavljeno strukturo shranimo neposredno v pripadajočo datoteko. Zaradi tega meni File ne vsebuje več nekaterih opcij za delo s projekti, ki smo jih bili uporabniki vajeni iz prejšnjih različic.

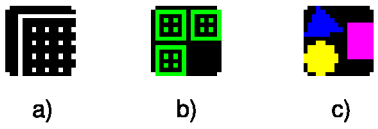
3.2.2 Meni Edit

Meni Edit vsebuje, kot je v navadi, opcije za razveljavljanje in uveljavljanje sprememb ter brisanje delov postavljene strukture, posebnost je le izbira *Layer Properties*. Orodje QCADesigner namreč podpira organiziranje kvantnih celičnih struktur v plasti (angl. *layers*). Prednost dela s plastmi je v tem, da lahko kvantne celice združujemo in na ta način manipuliramo z večimi naenkrat. Poleg tega pa QCADesigner vnaprej definira tri tipe plasti, ki se medseboj razlikujejo po funkcionalnosti:

- plast substrat (angl. *Substrate*),
- glavna plast celic (angl. *Main cell layer*) ter
- plast za risanje (angl. *Drawing layer*).

Tip substrat (slika 3.2a) služi predvsem kot mreža z vnaprej določenimi razmaki. Razmaki določajo položaje kvantnih celic pri postavljanju in s tem njihovo gostoto.

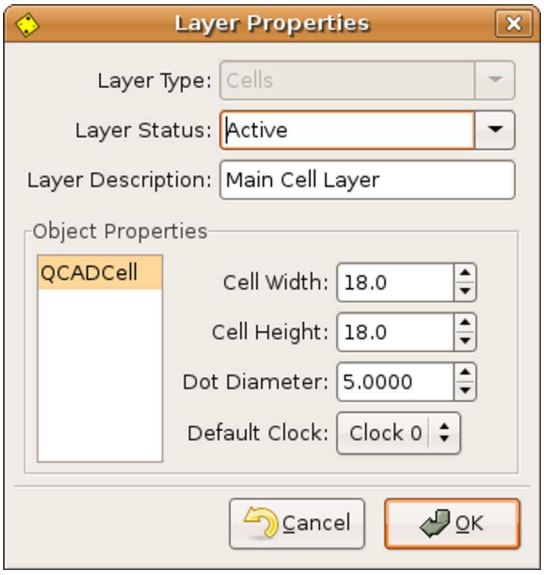
Za simulacijo delovanja postavljene strukture je najpomembnejši tip, imenovan glavna plast celic. Teh plasti je lahko poljubno mnogo in so v seznamu plasti označene z ikono s slike 3.2b. Vsebujejo celotno logično strukturo z vsemi pripadajočimi celicami ter oznakami vhodov in izhodov.



Slika 3.2 Ikone, ki v QCADesignerju označujejo posamezne tipe plasti: a) plast substrat, b) glavna plast celic in c) plast za risanje.

Zadnji tip plasti, ki ga podpira QCADesigner, je plast za risanje (slika 3.2c). Kot pove ime je namenjena raznim opombam, ki uporabniku dodatno pojasnjujejo delovanje posameznih delov postavljene strukture. Za simulacijo delovanja strukture ta plast ni pomembna, saj so oznake vhodov in izhodov strukture del glavnih plasti celic in se od tam tudi neposredno prenesejo v simulacijo.

Omenjena izbira *Layer Properties* torej omogoča natančnejše nastavljanje lastnosti posamezne plasti. Pogovorno okno, s katerim to počnemo, prikazuje slika 3.3. Prilagajati



Slika 3.3 Pogovorno okno z nastavitvami lastnosti posamezne plasti.

je mogoče naslednje parametre:

- *Layer Type*: to polje je že prednastavljeno in je odvisno od trenutne plasti. Uporabnik ga ne more spreminjati.
- *Layer Status*: določa ali je plast v delovnem področju vidna (angl. *Visible*) ali

Slika 3.4 Ikone za delo s plastmi ter dve pogosto uporabljani funkciji. Nahajajo se v vodoravni orodni vrstici in vodijo do naslednjih bližnjic: ikona a) omogoča ustvarjanje nove plasti, ikona b) odstranjevanje obstoječe plasti, ikona c) omogoča nastavljanje lastnosti plasti, ikona d) prikliče padajoči meni za izbiro plasti, ikona e) omogoča razvrščanje plasti, ikona f) prikliče padajoči meni za nastavljanje urinega področja, ikona g) pa omogoča grupiranje celic v vodilo.

nevidna (angl. *Hidden*) oz. ali naj jo simulacija upošteva (angl. *Active*).

- *Layer Description*: poljubno ime, ki ga uporabnik izbere za določeno plast.
- *Cell Width*: širina simuliranih kvantnih celic.
- *Cell Height*: višina simuliranih kvantnih celic.
- *Dot Diameter*: premer kvantne pike v simuliranih kvantnih celicah.
- *Default Clock*: privzeta ura za celice na določeni plasti, možno jo je kasneje spremninjati.

Pri delu s plastmi je potrebno upoštevati dejstvo, da QCADesigner ne podpira prenašanja *med* plastmi. Če je potrebno nek del strukture prenesti na drugo plast, ga je potrebno najprej na prvotni plasti izbrisati in ga nato znova postaviti na novi plasti. To seveda ni do uporabnika nič kaj prijazno in naj bi bilo v prihodnosti popravljeno.

Opozoriti je potrebno še na to, da QCADesigner ne podpira izrezovanja, kopiranja ali lepljenja (angl. *Cut, Copy, Paste*), ampak je to delno omogočeno z shranjevanjem (dela) strukture v posebno datoteko, ki jo kasneje lahko uvozimo kot del nove strukture (to je podrobneje opisano v opisu menija Tools).

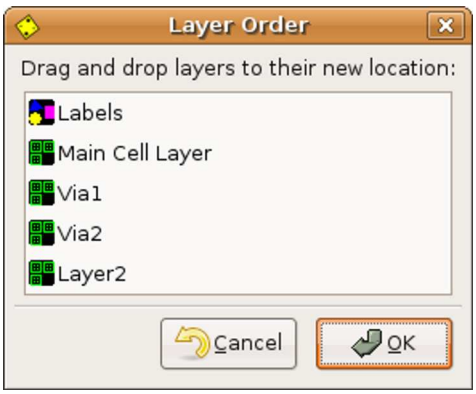
Ker je delo s plastmi zelo pogosto, se v vodoravni orodni vrstici nahajajo vse pomembnejše bližnjice do glavnih funkcij. Te so:

- ustvarjanje nove plasti v obstoječi postavitvi (angl. *Add Layer to Design*) (slika 3.4a),
- odstranjevanje plasti iz postavitve (angl. *Remove Layer from Design*) (slika 3.4b),
- nastavljanje lastnosti plasti (angl. *Layer Properties*) (slika 3.4c),

- padajoči meni za izbiro trenutne plasti in izbiro vidnosti posameznih plasti (slika 3.4d),
- razvrščanje plasti (slika 3.4e),
- padajoči meni za nastavljanje urinega področja (slika 3.4f) ter
- grupiranje celic v vodilo (slika 3.4g).

Zadnji dve bližnjici sicer nimata neposredne zveze s plastmi, vendar sta ti funkciji pogosto uporabljani in sta zato del orodne vrstice.

Ikona za razvrščanje plasti pravzaprav ni bližnjica, saj je funkcija razvrščanja dostopna le preko tega gumba. Po pritisku se prikaže pogovorno okno *Reorder Layers* (glej sliko 3.5), ki omogoča naknadno spreminjanje vrstnega reda obstoječih plasti. Plasti so naštete od najvišje ležeče do najnižje. Privzeto se na vrhu nahaja glavna plast celic. Vrstni red je mogoče preprosto spreminjati z vlečenjem imena plasti na drugo lokacijo, npr. plasti za risanje nad glavno plast celic.



Slika 3.5 Okno za urejanje vrstnega reda plasti.

3.2.3 Meni View

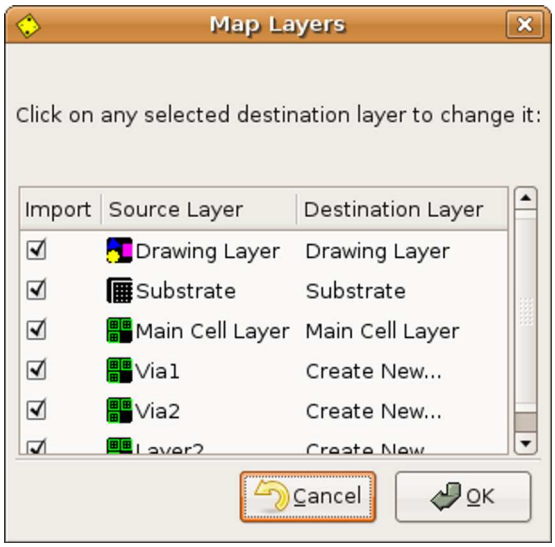
Meni View vsebuje opcije za prilagajanje pogleda na postavljeno strukturo. Vsebuje enake funkcije kot katerikoli grafični urejevalnik. Na prvi pogled je nova le izbira *Zoom Layer Extents*. Pa tudi ta izbira ni nič novega, le ime ni ravno v skladu z drugmi grafičnimi orodji. Klik nanjo povzroči prilagoditev prikaza vidnih plasti (angl. *visible*) tako, da so le-te v celoti vidne v delovnem področju, kar se običajno imenuje *Prilagodi oknu* (angl. *Fit to window*).

Naslednja dva menija pa je potrebno podrobneje opisati, saj vsebujeta glavne funkcije QCADesignerja.

3.2.4 Meni Tools

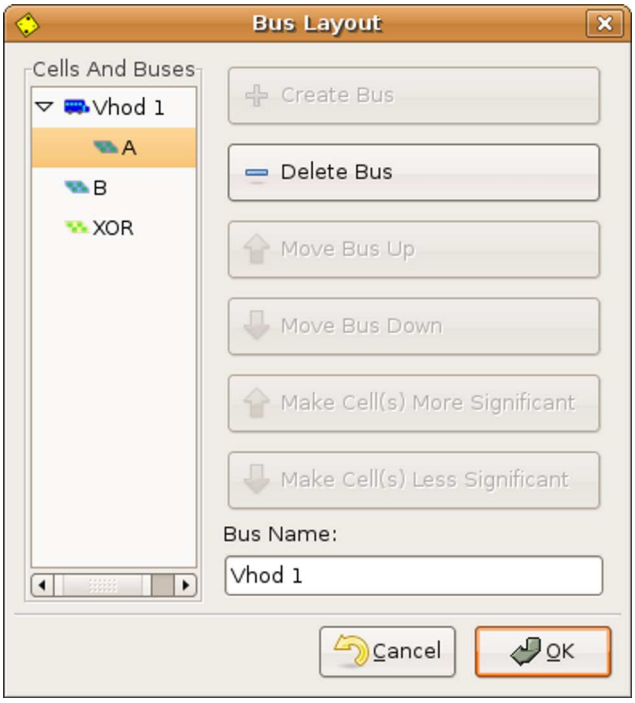
Meni Tools vsebuje orodja za delo s celicami v strukturi. Opciji *Create Block* in *Import Block* služita kot neke vrste trajnejše kopiranje in prilepljanje. Prva omogoča izvoz izbranih celic (te se imenujejo blok) v posebno datoteko za morebitno kasnejšo rabo. Tako je mogoče npr. shraniti del strukture, ki predstavlja zaključeno funkcionalno enoto, in je tako na voljo za uporabo v drugih strukturah. Celice, ki bodo sestavljale blok, je potrebno le označiti z miško in jih z izbiro *Create Block* shraniti kot nov blok v datoteko. Pri izbiri sta v veliko pomoč tipki *CTRL* in *SHIFT*, ki omogočata enostavnejše dodajanje in odstranjevanje celic iz izbora, podobno kot je to običajno v drugih grafičnih programih.

Opcija *Import Block* omogoča uvažanje predhodno shranjenih blokov. Uvoziti je mogoče tudi katerekoli predhodno shranjeno strukturo v celoti. Pri uvažanju je še potrebno ročno izbrati preslikavo plasti (slika 3.6), saj imajo običajno različne postavitve celic različno število plasti. Preslikava določa kako se bodo plasti iz uvoženega bloka spojile s plastmi v trenutni postavitvi. Določene plasti je mogoče pri uvažanju v celoti



Slika 3.6 Pogovorno okno z izbiro preslikave plasti.

izpustiti (npr. plast za risanje z dodatnimi opombami k strukturi), kar določa izbira *Import*. S tem je uporabniku omogočen uvoz le tistih delov struktur, ki ga zanimajo in je



Slika 3.7 Grupiranje večih celic ("signalov") v logično celoto.

kasnejšega ročnega brisanja celic in oznak kar najmanj. *Source Layer* se nanaša na plast v bloku, ki se uvaža, *Destination Layer* pa označuje ciljno plast v trenutni postavitvi. Tu je možno izbirati med preslikavo v že obstoječo ciljno plast in ustvarjanjem nove plasti v katero se bo preslikala izbrana plast bloka.

Naslednja zelo uporabna opcija je *Bus Layout*. Pri logičnem načrtovanju struktur je pogost primer, da neko informacijo sestavlja več signalov. Najenostavnejši tak primer so npr. seštevalniki, kjer je vsako število, zapisano v binarni obliki, predstavljeno z večimi biti, vsak bit pa ima neko fizikalno implementacijo. Tako npr. 4-bitno število na vhodu seštevalnika predstavljajo štiri vhodne celice, ki *skupaj* tvorijo število. Takemu grupiranju celic v *vodilo* je namenjena opcija *Bus Layout*. Le-ta omogoča, da več neodvisnih celic skupaj predstavlja neko logično enoto, npr. število. Tako ni potrebno v simulaciji iskati in preverjati stanja vsake celice posebej, ampak so stanja vseh, ki pripadajo istemu vodilu, izpisana skupaj v dvojiškem, desetiškem ali šestnajstiškem številskem sistemu. Grupiranje poteka s pomočjo pogovornega okna kot ga prikazuje slika 3.7. Do njega vodi tudi bližnjica v vodoravni orodni vrstici s slike 3.4g. Na levi strani pogovornega okna so prikazani vsi vhodi in izhodi, ki so trenutno definirani v postavitvi, in jih je mogoče

grupirati v vodilo. Z miško je potrebno izbrati želene vhode oz. izhode (tudi tu sta v pomoč tipki *CTRL* in *SHIFT*) in nato izbrati *Create Bus*. Vsako vodilo je mogoče tudi ustrezno poimenovati za boljšo preglednost simulacijskih rezultatov. Poleg ustvarjanja in brisanja vodil omogoča pogovorno okno še določanje vrstnega reda vodil in določanje pomembnosti posameznih celic vodila. Le-ta je pomembna za npr. pravilno desetiško predstavitev števil v simulacijskem poročilu.

Naslednji dve izbiri vplivata na dimenzije in postavitev celic. Prva, *Scale All Cells In Design*, omogoča spreminjanje fizičnih dimenzij vseh celic v postavitvi za določen faktor. Opozoriti je potrebno na dejstvo, da to ne spremeni samo pogleda na postavitev (povečava), ampak vpliva tudi na simulacijo delovanja. Druga opcija, *Rotate Selection*, pa omogoča vrtenje bloka celic v 90-stopinjskih korakih. To je posebej priročno pri uvažanju blokov celic, ki jih je potrebno nato prilagoditi in vključiti v obstoječo postavitev.

Predzadnja izbira v meniju Tools, izbira *Increment Cell Clocks*, omogoča upravljanje z uro (izbranim celicam je mogoče določiti posamezno fazo ure tudi preko bližnjice na vodoravni orodni vrstici; slika 3.4f). Ker ima ura štiri faze (kot je opisano v poglavju o osnovah kvantnih celičnih struktur), deluje ta izbira po modulu 4. Vsak klik na to opcijo poveča izbranim celicam fazo ure za eno fazo (izbrane celice imajo pri tem lahko pripisane različne faze ure).

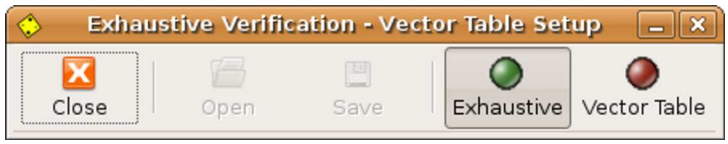
Zadnja izbira *Hide All But The Selected Layer* se nanaša na delo s plastmi. Omogoča da se uporabnik posveti delu na posamezni plasti tako da skrije ostale trenutno manj pomembne plasti. Po mojem bi bilo bolje da bi se ta opcija nahajala v meniju View, kjer so zbrane vse ostale izbire pogleda na postavitev.

3.2.5 Meni Simulation

Drug zelo pomemben meni v orodju QCADesigner je meni Simulation. Prvi dve izbiri, *Start Simulation* in *Stop Simulation*, služita za zagon in prekinitev simulacije. Ker simulacija delovanja večjih struktur traja kar precej časa, je gumb za prekinitev zelo koristen npr. v primeru, da je potrebno ta čas skrajšati s poenostavitvijo simulacijskih pogojev.

Izbira *Simulation Results* prikaže rezultate simulacije. Po končani simulaciji se le-ti prikažejo samodejno, zato je ta izbira uporabna predvsem za njihovo naknadno pregledovanje.

Naslednji izbiri v meniju Tools sta *Load Output From File* in *Save Output To File*. Namenjeni sta nalaganju predhodno shranjenih rezultatov iz datoteke in shranjevanju



Slika 3.8 Pogovorno okno, ki ponuja izbiro vrste simulacije.

rezultatov trenutne simulacije v datoteko. Opozoriti je potrebno na dejstvo, da se rezultati po končani simulaciji nahajajo le v delovnem pomnilniku računalnika in se sproti *ne* shranjujejo na trdi disk. Zato je dobro, posebej po daljših simulacijah, najprej shraniti rezultate in se šele nato posvetiti njihovi analizi. Program je namreč (še) v razvojni fazi in se zato lahko zgodi, da se med pregledovanjem rezultatov preneha odzivati ter ga je potrebno prisilno končati. S tem se seveda izgubijo tudi vsi rezultati simulacije in vsaj kanček uporabnikovega potrpljenja.

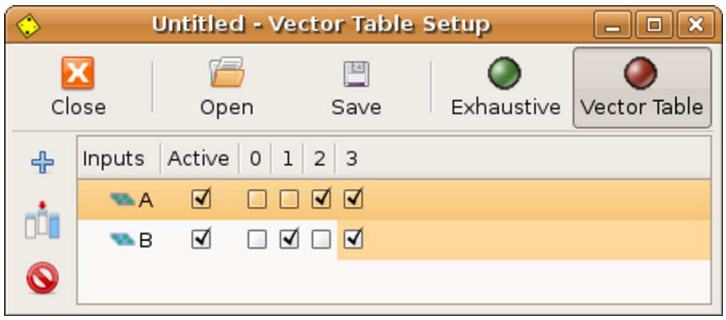
Zadnji izbiri omogočata dostop do parametrov simulacijskega pogona. Najprej je potrebno z izbiro *Simulation Type Setup* izbrati vrsto simulacije (slika 3.8). Na voljo sta naslednji:

- izčrpno preverjanje (angl. *Exhaustive Verification*) ter
- vektorska tabela (angl. *Vector Table*).

Privzeto je izbrano *izčrpno preverjanje*. Izčrpno preverjanje pomeni, da simulacijski pogon izračuna vse možne kombinacije vhodnih vektorjev in za vsakega izvede simulacijo delovanja strukture. To pomeni, da je pri n vhodih možnih 2^n vhodnih vektorjev, ki jih orodje razporedi po naraščajočem vrstnem redu, in za vsakega izračuna (izvede simulacijo) izhodni vektor. Simuliranje vseh možnih kombinacij najbolj temeljito testira postavljeno kvantno celično strukturo, vendar je tudi časovno najbolj zahtevno.

Včasih pa ni potrebno preveriti vseh možnih vhodnih kombinacij. Lahko se npr. zgodi, da se nekateri vhodni vektorji zaradi raznih omejitev nikoli ne pojavijo na vhodih. Takrat je odveč testirati kombinacije, ki ne bodo nikoli uporabljene, in s tem zapravljati čas. Mogoč je tudi primer, ko želi uporabnik preveriti delovanje strukture pri točno določenem zaporedju vhodnih vektorjev in ne po naraščajočem vrstnem redu, kot to naredi izčrpno preverjanje. Za take in podobne primere je na voljo simuliranje z *vektorsko tabelo*. To omogoča, da uporabnik ročno vnese vhodne vektorje, ki jih želi simulirati, ter poljubno nastavi vrstni red le-teh. Ta tip simulacije je na voljo samo takrat, ko so v postavljeni strukturi že definirani vhodi, sicer je onemogočen. Vnos vhodnih vektorjev

poteka preko pogovornega okna na sliki 3.9. V prvem stolpcu, poimenovanem *Inputs*, so



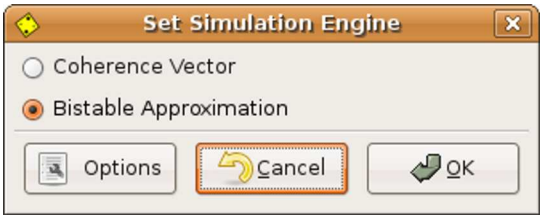
Slika 3.9 Vnos in določanje vrstnega reda vhodnih vektorjev za simulacijo z vektorsko tabelo.

našteti vsi vhodi v postavljeno strukturo. Imena vhodov so enaka imenom definiranim v strukturi. Če so definirana tudi vodila, so ustrezni vhodi predstavljeni v obliki vodil.

Drugi stolpec, stolpec *Active*, določa ali je nek vhod aktiven ali ne. Privzeta vrednost je aktiven. Če uporabnik določi, da je nek vhod neaktiven, ga simulacija ne obravnava več kot vhod z fiksno vrednostjo, ampak kot navadno celico. Z drugimi besedami, stanje celice je odvisno samo od stanja sosednjih celic. Na to je potrebno biti zelo pozoren, saj se lahko postavljena struktura zaradi te dodatne celice "obnaša" drugače, kot je bilo predvideno. Neaktivnost celice namreč ne pomeni, da bo simulacija potekala tako kot da bi to celico odstranili iz strukture, ampak ima celica tudi povraten vpliv na sosede.

Nove vektorje se definira s pomočjo gumbov na levi strani okna. Zgornji gumb doda nov vektor, srednji jih naknadno vrine med že definirane, spodnji pa jih briše. Definirani vektorji so prikazani kot oštevilčeni stolpci. Če so v strukturi definirana vodila, se pod zaporedno številko izpiše še desetiška vrednost vektorja, ki jo je mogoče tudi spreminjati in na ta način hitro postavljati bite na vhodih. Sicer pa se posamezne bite postavlja ali briše s pomočjo kljukic. Biti, ki imajo postavljeno kljukico, imajo vrednost logične enice, kjer pa kljukice ni, je vrednost postavljena na logično ničlo. Vpisovanje vrednosti vektorjev na tak način ni ravno hitro, zato je dobrodošla možnost shranjevanja in nalaganja vektorskih tabel iz datoteke. Orodju žal niso priložene nobene vnaprej pripravljene tabele, ki bi si jih uporabnik lahko prilagodil, tako da si jih mora prej ali slej vsak sam vnesti.

Ko so "podatki" za simulacijo pripravljeni, preostane le še izbira *simulacijskega pogona* (algoritma) in nastavitev pripadajočih parametrov z izbiro opcije *Simulation Engine Setup* v meniju Simulation (slika 3.10). QCADesigner ima v novejših verzijah vgrajena



Slika 3.10 Izbira simulacijskega pogona (algoritma).

dva simulacijska pogona (v starejših so bili štirje). To sta:

- koherenčni vektor (angl. *coherence vector*) ter
- bistabilna aproksimacija (angl. *bistable aproximation*).

Oba pogona in nabora parametrov (gumb *Options*) sta natančneje opisana v podpoglavju 3.3.

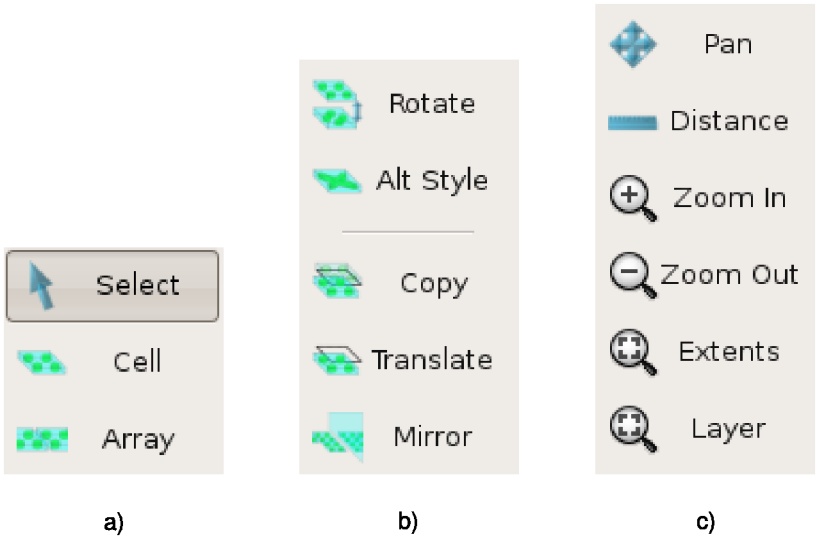
3.2.6 Meni Help

Zadnji meni v orodju QCADesigner je meni Help, ki (kot je v navadi) vsebuje vgrajeno pomoč pri uporabi programa. Pomoč je dokaj skopa in vsebuje le najnujnejše informacije za uporabo programa, zato je potrebno precej poskušanja in ugibanja preden uporabnik dobro spozna zmogljivosti in omejitve programa.

3.2.7 Orodna vrstica

Medtem ko so funkcije, ki se nahajajo v menijih, namenjene podrobnejšem nastavljanju orodja in struktur, poteka postavljanje kvantne celične strukture skoraj izključno s pomočjo navpične orodne vrstice (slika 3.11). Le-ta vsebuje nabor orodij za ustvarjanje in upravljanje s celicami in prilagajanje pogleda na strukturo. Nabor se nekoliko spreminja v odvisnosti od trenutno izbrane plasti, vendar te razlike niso velike.

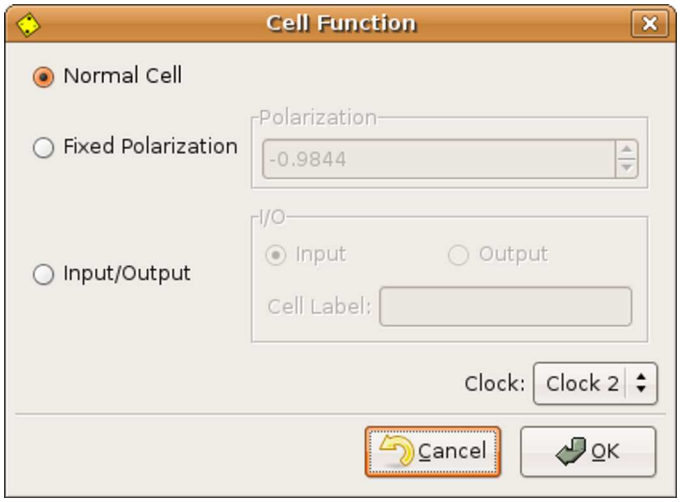
Kot je običajno za grafične programe, je prvo orodje, ki je ponujeno v orodni vrstici, orodje za izbiranje elementov na zaslonu – v tem primeru so to QCA celice, orodje pa se imenuje *Select*. Izbirati je mogoče več celic naenkrat z držanjem levega gumba miške in vlečenjem pravokotnika preko željenih celic. Izbrane celice se obarvajo rdeče. Dodatne celice je mogoče dodati k že zaključenemu izboru z držanjem tipke *SHIFT*. Posamezne celice ali skupine celic se lahko iz izbora odstrani z držanjem tipke *CTRL* in ponovnim izbiranjem odvečnih celic.



Slika 3.11 Navpična orodna vrstica. Orodja se delijo na tri skupine: a) orodja za gradnjo strukture, b) orodja za delo s celicami ter c) orodja za upravljanje s pogledom na strukuro.

Izbiranje celic je pomembno, ker je le na tak način mogoče nastavljati funkcije celic (angl. *cell functions*). Potrebno je le dvakrat klikniti na celico oz. izbor celic in prikaže se pogovorno okno na sliki 3.12. V starejših verzijah QCADesignerja je bil dostop do njega mogoč tudi preko menijev, v novejših pa tega ni več, kar nekoliko zmede uporabnike, vajene dela s starejšimi različicami. Kot je razvidno iz slike, loči QCADesigner tri tipe celic:

- navadna celica (angl. *normal cell*): ta nima vnaprej definirane funkcije in spreminja stanje le na podlagi vplivov sosednjih celic;
- celica s fiksno polarizacijo (angl. *fixed polarization*): taki celici uporabnik vnaprej pripiše neko polarizacijo, v kateri celica ostane celoten čas simulacije, ne glede na vplive sosednjih celic in ure; polarizacija je določena z realnim številom iz intervala $[-1, 1]$, ki ga je potrebno vpisati v vnosno vrstico z imenom *Polarization*;
- vhodna oz. izhodna celica (angl. *input/output cell*): vhodnim celicam določa polarizacijo simulacijski pogon v odvisnosti od izbrane vrste simulacije; pri izčrpnem preverjanju tako na vhodih nastopijo vse možne kombinacije vhodnih vektorjev, pri simulaciji z vektorsko tabelo pa le uporabniško določeni vektorji; izhodne celice so v bistvu navadne celice s to razliko, da se njihovo stanje prikazuje v grafih, ki



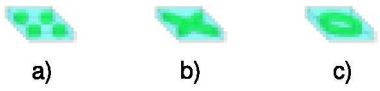
Slika 3.12 Pogovorno okno za določanje funkcij celic.

podajajo rezultate simulacije; stanja preostalih celic v rezultatih simulacije namreč niso izpisana; zaradi izpisovanja stanj vhodnih oz. izhodnih celic v poročilu, je potrebno pri tem tipu v vnosno polje *Cell Label* vpisati še ime vhoda oz. izhoda; pametno izbrana imena zelo olajšajo analizo obsežnega simulacijskega poročila.

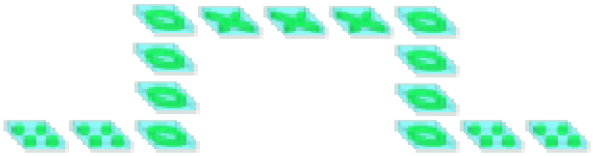
V istem pogovornem oknu je, poleg funkcije, mogoče izbranim celicam nastaviti še uro, kar je sicer dosegljivo tudi neposredno na vodoravni orodni vrstici.

Pri določanju funkcije celic velja opozoriti še na eno posebnost QCADesignerja. Dokaj pogosto se dogaja, da uporabnik želi, da se določene celice pred začetkom simulacije nahajajo v vnaprej definiranem stanju, kasneje pa se lahko to stanje spreminja v skladu z vplivi sosednjih celic, ure, itd. Tak tip celic se pojavlja v skoraj vsaki večji kvantni celični strukturi. V QCADesignerju je mogoče tako celico ustvariti le tako, da uporabnik najprej določi celico kot celico s fiksno polarizacijo, nato pa ponovno spremeni tip celice v navadno celico. V prikazu postavljene strukture nato taka celica izgleda kot celica s fiksno polarizacijo, le drugačne barve. Težava je le v tem, da omenjena funkcionalnost ni nikjer dokumentirana in sem porabil precej časa, da sem ugotovil kako take celice ustvariti, saj brez njih določene strukture ne delujejo pravilno.

Naslednji dve orodji, *Cell* in *Array*, služita dejanskemu postavljanju strukture. Prvo je namenjeno postavljanju posameznih celic v strukturo, drugo pa postavljanju večjega števila celic obenem. Orodje poskrbi za to, da se dve ali več celic ne more medsebojno prekrivati, če na primer skuša uporabnik napačno postaviti celice.



Slika 3.13 Stili celic, kot jih definira QCADesigner: a) normalni, b) nadvozni in c) navpični stil.



Slika 3.14 "Nadvoz", ki omogoča nemoten prehod ene linije preko druge.

Orodje *Rotate* je namenjeno rotiranju celic okrog središča posamezne celice. V bistvu se spreminja položaj kvantnih pik znotraj kvantne celice s 45-stopinjskim korakom. Tako se lahko kvantne pike namesto v vogalih "kvadrata" nahajajo na razpoloviščih stranic, ohranijo pa se medsebojne razdalje med pikami. Če je predhodno izbranih več celic, se rotirajo pike v vseh hkrati z enakim korakom. Velja opozoriti, da to orodje ni ekvivalentno izbiri *Rotate Selection* v meniju Tools, ki obrača izbran del strukture okrog središča strukture.

Orodje *Alt Style* ni povezano z spreminjanjem fizikalnih lastnosti celic, ampak spreminja le njihov prikaz, celice pa so s stališča simulacije ekvivalentne. Definira tri različne stile prikazovanja celic:

- normalni (angl. *normal*) – slika 3.13a,
- nadvozni (angl. *crossover*) – slika 3.13b ter
- navpični stil (angl. *vertical*) – slika 3.13c.

Privzeto je izbran normalni stil. Ostala dva sta namenjena predstavitvi prehodov ene linije preko druge v dveh različnih plasteh. V perspektivi prikazuje tak prehod slika 3.14. Večinoma se to uporablja pri simulaciji delovanja večplastnih kvantnih celičnih struktur (vezji), kjer je potrebno povezati večje dele segmente, pri čemer se vsak nahaja na svoji plasti. Po dogovoru tvorijo celice prikazane z navpičnim stilom "podporne stebre", celice z nadvoznim stilom pa "nadvoz" – od tod tudi ime. "Nadvoz" se nahaja na plasti, običajno poimenovani *Crossover*, podporniki pa na posebnih plasteh *Via*. Leteh je lahko v principu poljubno mnogo, v praksi pa sta običajno dve. Razdaljo med

plastmi je mogoče spreminjati z lastnostjo *Layer Separation*, ki se nahaja v nastavitvah simulacijskega pogona.

Ker so velike strukture pogosto sestavljene iz večjega števila enakih oz. podobnih delov, je nesmiselno risati vsakega znova. Tu priskoči na pomoč orodje *Copy*, ki izdela kopijo trenutnega izbora celic. Kopija se pojavi na istem mestu, kjer se nahaja original in jo je potrebno ročno povleči na željeno mesto.

Natančno pozicioniranje izbora celic je mogoče z orodjem *Translate*. Privzeto se namreč celic ne da premikati poljubno, ampak se samodejno poravnavajo glede na predefinirano koordinatno mrežo (angl. *Snap To Grid*). Orodje *Translate* pa omogoči premikanje izbora celic za poljubno število nanometrov, neodvisno od mreže.

Zadnje orodje za rokovanje s celicami je orodje *Mirror*. Omogoča vertikalno ali horizontalno zrcaljenje izbranih celic. To je lahko precej uporabno v primerih, ko ima struktura simetrično zgradbo, kot npr. D-pomnilna celica. V tem primeru je potrebno postaviti le del strukture, simetričen del pa je preprosto postavljen z uporabo orodij *Copy* in *Mirror*. Potrebno je le biti pozoren na uro, saj ni nujno, da po zgradbi enaki deli uporabljajo tudi isto uro. Običajno je ravno nasprotno in je potrebno uro posameznim celicam ponovno določiti, kar pa lahko uporabnik, posebej pri večjih strukturah, hitro spregleda.

Naslednja orodja v navpični orodni vrstici so namenjena prilagajanju pogleda na postavljeno strukturo. Najpogosteje uporabljeno je orodje *Pan*. To uporabniku omogoča, da strukturo v glavnem oknu "zgrabi" z miško in jo poljubno premika. Pri tem se premika celotna postavitev s koordinatnim sistemom vred, kar omogoča hitro premikanje in pregledovanje.

Orodje *Distance* je namenjeno merjenju razdalj med poljubnimi točkami v strukturi. Razdalje je možno meriti le vodoravno ali navpično, ne pa tudi diagonalno. Merjenje je aktivno, dokler je pritisnjena tipka na miški. Med merjenjem se izrisuje ravnilo z nanometrsko skalo s pomočjo katerega lahko uporabnik hitro odčita razdaljo. Na ta način je mogoče enostavno preveriti ali so posamezne celice oz. skupine celic dovolj narazen, da med njimi ne prihaja do neželenih vplivov.

Nadalje sta na voljo orodji *Zoom In* in *Zoom Out*, ki imata enako vlogo kot podobna orodja v grafičnih programih. Z njima si lahko uporabnik pobližje ogleda podrobnosti postavljene strukture ali pa nastavi pogled tako, da vidi celotno strukturo. To je predvsem uporabno pri preverjanju pravilnosti delovanja, pa tudi *med* samo simulacijo.

Pri izbiri parametrov simulacije je namreč med drugim na voljo tudi opcija *Animate*, ki omogoča da uporabnik opazuje dogajanje v strukturi (perturbacijo polarizacije celic) med simulacijo.

Še zadnji orodji v navpični orodni vrstici sta *Extents* in *Layer*. Imeni sta nekoliko neposrečeno izbrani, saj nista v skladu s podobnimi orodji v grafičnih programih. Prvo je namenjeno prilagoditvi pogleda tako, da je obenem vidna celotna struktura in izkoriščena celotna delovna površina. Ta opcija je navadno bolj znana pod imenom *Prilagodi oknu* (angl. *Fit to window*). Drugo orodje pa ima podobno vlogo, le s to razliko, da prilagodi pogled velikosti trenutno izbrani plasti (angl. *Fit layer to window*).

To so vsa orodja, ki so na voljo za delo na plasteh, ki vsebujejo kvantne celice. Pri delu na plasti z oznakami pa se nabor orodij spremeni. Onemogočena so orodja za rokovanje s celicami, na voljo pa je novo orodje *Labels*. Kot pove ime, je to namenjeno dodajanju informacijskih oznak k poljubnim delom strukture.

S tem je zaključen pregled funkcij, ki omogočajo postavljanje in prilagajanje struktur v QCADesignerju. V naslednjem razdelku sta podrobneje opisana simulacijska pogona, ki sta srce orodja in hkrati povezava med simulacijo in njenim fizikalnim ozadjem.

3.3 Simulacijski pogon

Kot je bilo omenjeno v prejšnjem razdelku ima QCADesigner dva simulacijska pogona, ki se med seboj razlikujeta po hitrosti in natančnosti simulacije delovanja QCA "vezja". Nabor parametrov je pri obeh precej različen, zato si velja natančneje pogledati delovanje vsakega izmed njiju.

3.3.1 Koherenčni vektor

Koherenčni vektor (angl. *coherence vector*) sloni na vektorski interpretaciji matrike gostote ρ posamezne kvantne celice ob pomoči Paulijevih spinskih matrik[4] σ_x, σ_y in σ_z. Natančnejša fizikalna izpeljava simulacijskega algoritma tu ne bo podana, saj je bolj zanimiva za razvijalce orodja kot za končnega uporabnika. Izpeljavo si lahko uporabnik pogleda v dokumentaciji QCADesignerja, kjer je tudi za novopečenega uporabnika dokaj razumljivo razložena.

Za lažjo predstavo o časovni zahtevnosti algoritma, pa je potrebno omeniti, da ta simulacijski pogon uporablja parcialne diferencialne enačbe, ki opisujejo dogajanje v po-

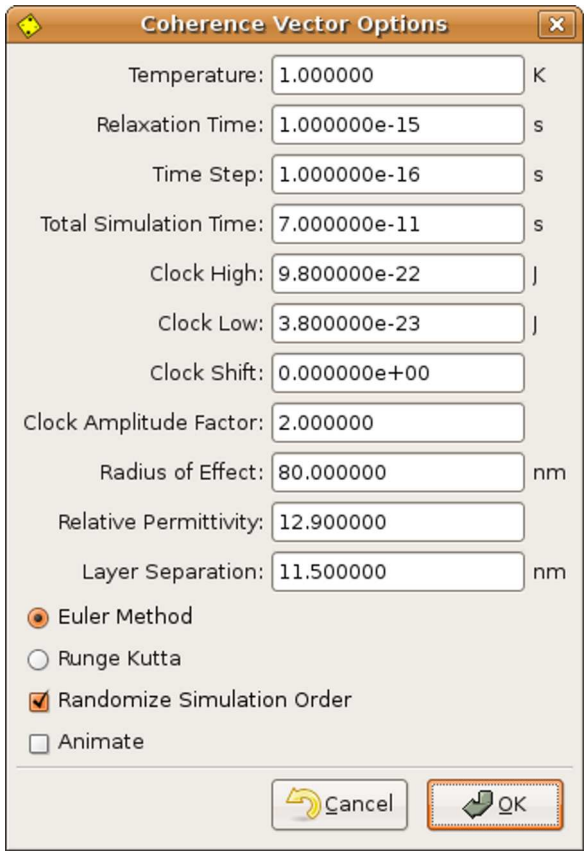
[4] http://www.qcadesigner.ca/manual/index.html

samezni celici. Celico modelira kot enostaven sistem z dvema možnima stanjema. Za vsak časovni korak in za vsako celico posebej najprej izračuna vektor energije okolice $\vec{\Gamma}$ (angl. *Energy environment*), ki opisuje vpliv okolice na izbrano celico, nato pa še koherenčni vektor stabilnega stanja $\vec{\lambda}_{ss}$ (angl. *Steady state coherence vector*), ki je neposredno odvisen od energije okolice $\vec{\Gamma}$. Ko je koherenčni vektor za vsako celico izračunan, se algoritem pomakne za en časovni korak naprej in ponovi izračune.

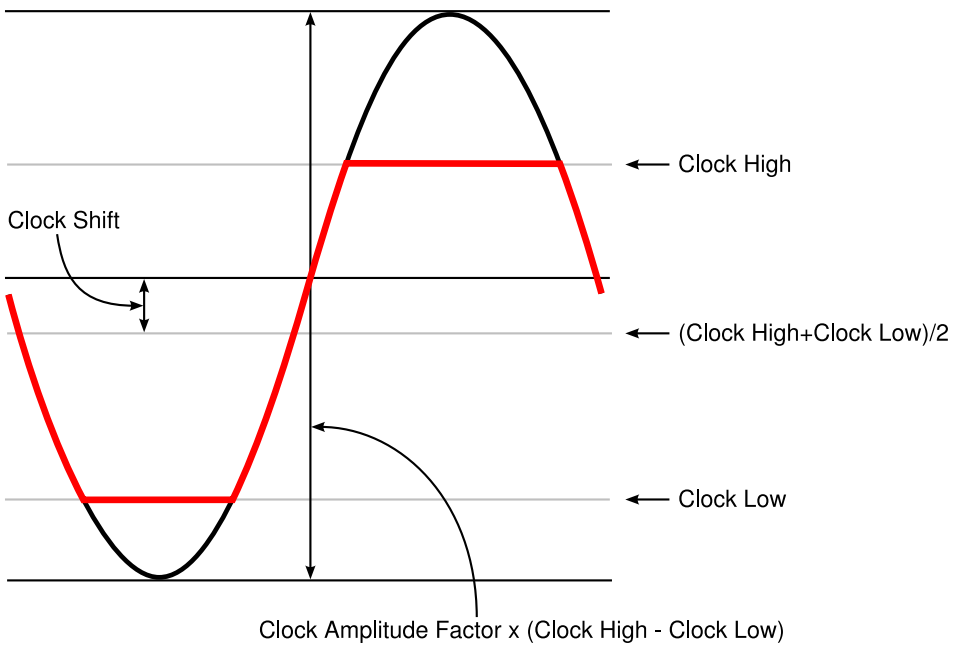
Ker algoritem vključuje računanje diferencialnih enačb, je časovno zelo zahteven. Očitno je, da je z večjim številom celic zahtevnost vse večja, saj je število diferencialnih enačb neposredno odvisno od števila celic. Zato je ta pogon precej počasnejši od bistabilne aproksimacije, ki uvaja določene poenostavitve glede delovanja celic. Njegova dobra stran pa je, da dobro simulira dinamiko dogajanja v celicah.

Za končnega uporabnika, ki se ne ukvarja s fizikalnimi zakonitostimi metod simuliranja, je bolj kot algoritem zanimiv nabor parametrov, katere lahko prilagaja svojim potrebam. Nabor je prikazan na sliki 3.15, ki prikazuje pogovorno okno QCADesignerja za spreminjanje parametrov (do njega vodi gumb *Options* v pogovornem oknu s slike 3.10). Parametri so naslednji:

- **temperatura** (angl. *temperature*): temperatura v stopinjah Kelvina pri kateri poteka simulacija; izbrana temperatura vpliva na osnovno stanje celic;
- **relaksacijski čas** (angl. *relaxation time*): konstanta, ki opisuje disipacijo energije v okolico; določa kako hitro sistem doseže osnovno stanje;
- **časovni korak** (angl. *time step*): čas, ki preteče med dvema korakoma; če je izbran predolg čas, lahko postane zaradi narave algoritma simulacija nestabilna in se samodejno konča; če je ta čas v primerjavi s celotnim časom simulacije (angl. *total simulation time*) prekratek, lahko simulacija traja predolgo za praktično uporabo (več dni!);
- **celoten čas simulacije** (angl. *total simulation time*): velja opozorilo iz prejšnje alineje; ta parameter skupaj s časovnim korakom definira pojem *koraka*; celotno število korakov je namreč definirano kot kvocient med celotnim časom simulacije in časovnim korakom;
- **visoko** in **nizko stanje ure** (angl. *clock high, clock low*), **amplituda** (angl. *clock amplitude factor*) ter **odmik** (angl. *clock shift*): ti parametri skupaj določajo obliko



Slika 3.15 Nabor parametrov, ki jih je mogoče prilagajati pri simulaciji s koherenčnim vektorjem.

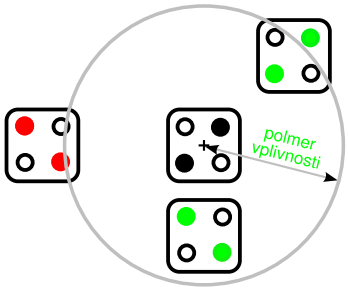


Slika 3.16 Parametri, ki določajo obliko urinega signala. Rdeča krivulja ponazarja dejansko obliko urinega signala, ki jo uporablja simulacijski pogon.

urinega signala na način kot ga prikazuje slika 3.16; urin signal je namreč izračunan s pomočjo sinusoide in neposredno vpliva na višino pregrad med kvantnimi pikami v celici; parametra visoko in nizko stanje določata energijo nasičenja urinega signala, medtem ko omogoča pozitiven ali negativen odmik premikanje signala po vertikali; amplituda se pomnoži z razliko med visokim in nizkim stanjem in torej določa amplitudo sinusoide, s katero je urin signal modeliran;

- **polmer vplivnosti** (angl. *radius of effect*): vpliv neke celice na njeno sosedo pada s peto potenco razdalje med njima[5]; vpliv ene celice na drugo torej postane na nekoliko večji razdalji zanemarljiv in algoritem je mogoče nekoliko pohitriti; ni namreč potrebno računati vpliva vsake celice v strukturi na vse ostale, ampak je dovolj upoštevati le tiste v bližnji okolici; polmer vplivnosti torej določa, katere sosede neke izbrane celice bodo še upoštevane kot take z nezanemarljivim vplivom; upoštevajo se vse, ki imajo središče znotraj kroga, ki ga določa polmer vplivnosti (glej sliko 3.17); če je polmer zelo majhen in zajame le najbližje celice, je simulacija

[5] http://www.qcadesigner.ca/manual/index.html



Slika 3.17 Polmer vplivnosti, ki določa katere sosede celice imajo nazanemarljiv vpliv. Celice, ki imajo kvantne pike obarvane zeleno, imajo na centralno celico nezanemarljiv vpliv, saj se njihova središča nahajajo znotraj polmera vplivnosti. Vpliv celice z rdečimi kvantnimi pikami pa se zanemari.

delovanja nenatančna in rezultati niso praktično uporabni; če so postavljene strukture večplastne, ta polmer določa kroglo in omogoča, da so upoštevani tudi vplivi celic na drugih plasteh (seveda le v primeru, da je razmak med plastmi manjši od polmera vplivnosti);

- **relativna dielektričnost** (angl. *relative permittivity*): ta parameter je karakteristika materiala, v katerem se simulirane celice nahajajo in se uporablja za računanje *perturbacijske energije* (energija, ki je potrebna, da imata dve celici nasprotno polarizacijo in je odvisna od elektrostatične interakcije med naboji[6]); prednastavljena vrednost ustreza materialu GaAs oz. AlGaAs;

- **razmak med plastmi** (angl. *layer separation*): določa fizično razdaljo med dvema plastema celic v večplastnih strukturah; podana je v nanometrih;

- **Eulerjeva metoda/metoda Runge-Kutta** (angl. *Euler method, Runge Kutta method*): za reševanje diferencialnih enačb, ki so osnova algoritma, lahko uporabnik izbere eno od znanih metod za numerično reševanje;

- **naključni vrstni red simulacije** (angl. *randomize simulation order*): določa naključni vrstni red pri izračunu novega stanja celic znotraj posamezne iteracije algoritma; avtorji priporočajo, da je ta opcija omogočena;

- **animacija** (angl. *animate*): omogoča da lahko uporabnik med simulacijo opazuje dogajanje v kvantnih celicah, saj se polarizacije izrisujejo v skladu s trenutnim

[6] http://www.qcadesigner.ca/manual/index.html

korakom simulacije; to precej upočasni izvajanje simulacije in je primerno le za manjše strukture, kjer je s pogledom še mogoče slediti spremembam; pri takem načinu opazovanja vezja bi bila zelo koristna možnost začasnega ustavljanja simulacije oz. izvajanja po časovnih korakih, saj je sicer številnim spremembam težko slediti, če pa uporabnik simulacijo prekine, se nato ta prične ponovno od začetka.

3.3.2 Bistabilna aproksimacija

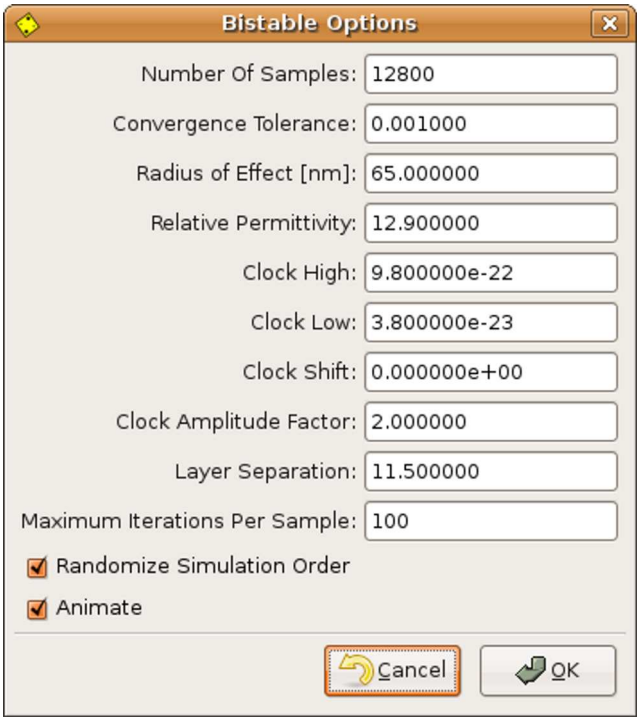
Podobno kot prejšnji simulacijski pogon tudi *bistabilna aproksimacija* (angl. *bistable aproximation*) modelira posamezno celico kot enostaven sistem z dvema možnima stanjema. Tudi tu sledi le okviren opis delovanja algoritma, fizikalno ozadje pa je podrobneje razloženo v pomoči[7] orodja QCADesigner.

Za razliko od koherenčnega vektorja bistabilna aproksimacija operira z vsoto energij iz okolice, ki vpliva na delovanje posamezne celice. Pogon na podlagi Schrödingerjeve časovno neodvisne enačbe izračuna osnovno stanje celice. Za računanje lastnih vrednosti (energije, vezane na stanje celice) in lastnih vektorjev (stanje celice) hamiltonke sistema z dvema stanjema, se uporablja Jakobijev algoritem. Algoritem nato vsa dobljena stanja celice razvrsti tako, da je na prvem mestu tisto, kateremu ustreza najmanjša energija. Računanje stanj celic se ponavlja do takrat, ko celotna struktura konvergira v vnaprej določene tolerančne okvirje. Takrat se shrani stanja izhodov in simulacija se ponovi za nove vhodne vrednosti.

Avtorji orodja opozarjajo, da je bistabilna aproksimacija namenjena predvsem preverjanju logične pravilnosti strukture, ne opisuje pa najbolje dinamike dogajanja v celicah (za to je veliko boljši koherenčni vektor). Najpočasnejši korak algoritma je razvrščanje stanj vseh celic, vendar je to še vedno veliko hitrejše kot računanje diferencialnih enačb pri simulaciji s koherenčnim vektorjem. Zato je bistabilna aproksimacija hitrejša in omogoča simuliranje delovanja struktur z večjim številom celic v praktično sprejemljivem času. Zaradi tega je tudi zelo primerna za sprotno preverjanje pravilnosti delovanja strukture med postavljanjem, ko pa je le-ta enkrat končana, je smiselno izvesti podrobnejšo (in dolgotrajnešo) simulacijo s koherenčnim vektrojem.

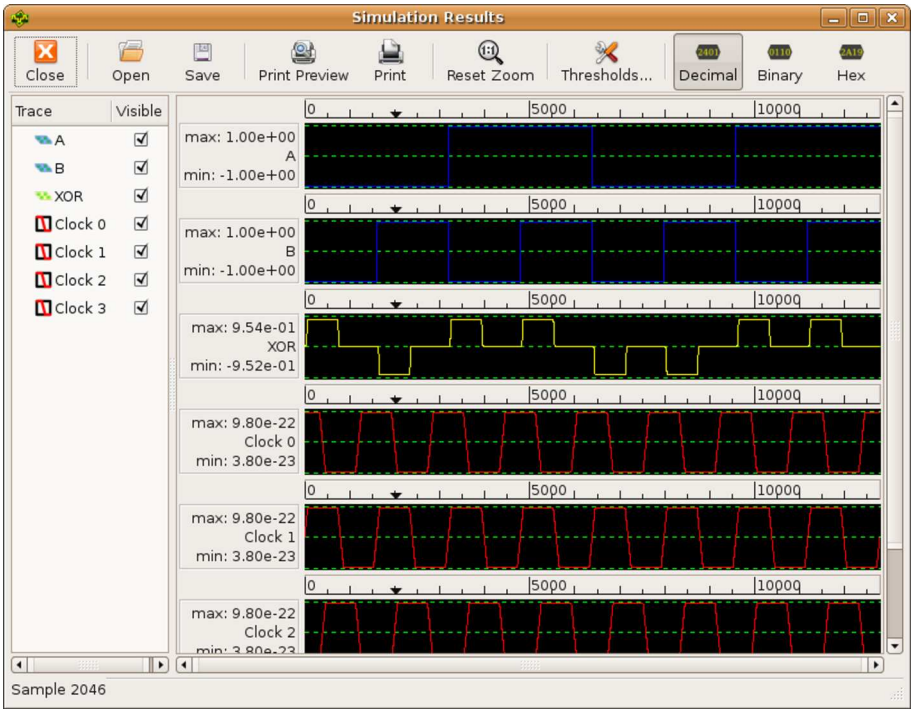
Nabor parametrov, ki jih je mogoče spreminjati, je podoben tistemu pri simulaciji s koherenčnim vektorjem, zato so podrobneje opisani le parametri, ki se razlikujejo (glej tudi podpoglavje 3.3.1 in sliko 3.18):

[7] http://www.qcadesigner.ca/manual/index.html



Slika 3.18 Nabor parametrov, ki jih nudi simulacija z bistablino aproksimacijo.

- **število korakov** (angl. *number of samples*): celotna simulacija je sestavljena iz števila korakov, ki ga določa ta parameter; za vsak korak posebej simulacijski algoritem izračuna polarizacijo vseh celic; pri tem upošteva vplive sosednjih celic; iz tega sledi, da večje število vzorcev pomeni natančnejšo simulacijo, hkrati pa tudi večjo časovno zahtevnost; če je vrednost tega parametra prenizka, so lahko rezultati simulacije v nasprotju s pričakovanji, saj bo na razpolago premalo korakov med spreminjanjem vhodov; pri privzetih vrednostih ostalih parametrov zato avtorji priporočajo da je število vzorcev pri uporabi vektorske tabele enako $1000 \times N$ (N je število vhodnih vektorjev) oz. 2000×2^n (n je število vhodov), pri izčrpnem preverjanju;
- **konvergenčna toleranca** (angl. *konvergence tolerance*): določa dovoljeno odstopanje od konvergence; algoritem računa polarizacijo celic znotraj enega koraka toliko časa, dokler ne postane sprememba polarizacije poljubne celice manjša od konvergenčne tolerance;
- **največje število iteracij/vzorec** (angl. *maximum iterations per sample*): določa



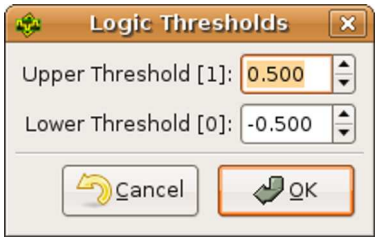
Slika 3.19 Orodje za pregledovanje rezultatov simulacije.

največje število iteracij v posameznem koraku; v teh mora struktura doseči konvergečno stanje; če konvergenca ni dosežena, se simulacija kljub temu premakne na naslednji korak; za kompleksnejše strukture mora biti to število večje.

3.4 Pregledovanje rezultatov simulacije

Poleg algoritmov za simuliranje delovanja kvantne celične strukture nudi QCADesigner tudi orodje za pregledovanje in analizo rezultatov simulacije (glej sliko 3.19). Okno sestavljajo trije deli – vodoravna orodna vrstica, ki vsebuje spodaj opisane gumbe, okno kjer so našteti vhodni in izhodni signali oz. vodila ter okno z grafi poteka vseh naštetih signalov oz. vodil.

Pregledovalnik omogoča prikazovanje in shranjevanje rezultatov trenutne simulacije (gumb *Save*) ter nalaganje rezultatov preteklih simulacij iz datoteke (gumb *Load*). To pride zelo prav pri rezultatih simulacije delovanja obsežnejših struktur, kjer analiza navadno traja dlje časa. Omogoča tudi "tiskanje" grafov v datoteko *PostScript* (gumb *Print*), podobno kot je to opisano pri tiskanju struktur. Podobno tudi tu gumb *Print Preview*



Slika 3.20 Pogovorno okno za nastavljanje pragov logičnih nivojev.

(še) ne deluje.

Posamezen graf poteka signala, ki se v QCADesignerjevi terminologiji imenuje *sled* (angl. *trace*), si je mogoče tudi natančneje ogledati oz. približati. Z miško je potrebno narisati okvir okoli področja, ki nas zanima. To se nato samodejno poveča in razkrije morebitne oscilacije signala. Na prvoten pogled se je mogoče vrniti z gumbom *Reset Zoom*.

Gumb *Thresholds* služi za nastavljanje pragov logičnih nivojev (slika 3.20). Določa od katere vrednosti dalje se vrednost signala interpretira kot logična enica (angl. *upper treshold*) in pod katero vrednostjo kot logična ničla (angl. *lower threshold*). To vpliva na desetiško, dvojiško ali šestnajstiško predstavitev vrednosti signala, med katerimi je mogoče izbirati z gumbi *Decimal*, *Binary* in *Hex*. Taka predstavitev zelo pohitri analizo signalov, saj uporabniku ni potrebno opazovati vsakega signala v vodilu posebej, ampak lahko preveri vse signale obenem s pomočjo npr. njihove desetiške vrednosti. To je posebej priročno pri npr. preverjanju izhoda števcev.

Na levi strani okna, kjer so našteti vsi signali, je mogoče z opcijo *Visible* izbirati med vidnostjo posameznih signalov. Tako lahko uporabnik skrije vse nepomembne signale in hkrati tudi pohitri premikanje po grafih. Poleg vhodov in izhodov, ki so definirani v postavljeni strukturi, so naštete tudi vse štiri faze ure in desno prikazane tudi njihove sledi. To je nujno, saj so stanja signalov odvisna tudi od ure in uporabnik mora vedeti v kateri fazi ure mora opazovati vrednost signala.

Kot rečeno, glavnino okna zavzema prikaz sledi signalov. Na vodoravni osi grafa se izpisuje število korakov. Trenutna številka koraka pod kurzorjem se prikazuje v statusni vrstici pod grafi. Navpična os pa prikazuje vrednost signala v določenem koraku. Mejne vrednosti vhodnih in izhodnih signalov ter ure je mogoče nastaviti v parametrih simulacije.

S tem je zaključen opis funkcij in zmožnosti orodja QCADesigner. Praktična uporaba na zahtevnejših primerih kvantnih celičnih struktur pa je opisana v naslednjem poglavju.

4 Postavitve aritmetično-logičnih struktur

4.1 Uvod

Prejšnje poglavje je podrobno opisalo funkcionalne zmožnosti in omejitve orodja QCADesigner. Opozorilo je tudi na nekatere nedodelanosti in napake, ki na začetu zagrenijo marsikatero uro dela s programom. Vendar pa nobena stvar ni popolna in bistveno je, da se zna človek prilagoditi in kar najbolje izkoristiti ponujene funkcije. Zato bo v tem poglavju podanih nekaj kompleksnejših primerov postavitve in simulacije kvantnih celičnih struktur s QCADesignerjem. Vsi predstavljeni primeri so s področja *aritmetično-logičnih struktur*, saj je aritmetika s števili v fiksni vejici osnovni način za reševanje problemov na računalnikih. Z njo je mogoče realizirati tudi operacije v plavajoči vejici, kar se pogosto uporablja na cenejših mikroprocesorjih. Podobno je tudi z operacijami nad nenumeričnimi operandi, kjer je najpogosteje uporabljana operacija primerjave dveh operandov. To je prav tako mogoče realizirati z odštevanjem števil v fiksni vejici. Iz povedanega sledi, da je aritmetično-logična enota eden izmed najvažnejših delov centralne procesne enote vsakega računalniškega sistema in bistveno vpliva na njegove performančne parametre ter ceno. Zato je vedno predmet optimizacij v hitrosti delovanja, velikosti, ceni

izdelave, ipd. Teoretično imajo kvantne celične strukture na vseh teh področjih veliko potencialno prednost pred klasičnimi tranzistorskimi realizacijami, zato je smiselno raziskati kako optimalne oz. "nerodne" so kvantne celice pri bolj praktični uporabi v logičnih struktruah.

Večina predstavljenih struktur je postavljenih po analogiji s klasičnimi tranzistorskimi realizacijami. To pomeni, da imajo predstavljene kvantne celične in klasične strukture na nivoju posameznih logičnih elementov (AND-, OR-vrata, D-pomnilne celice, itd.) *enako zgradbo*, *različne* pa so si *po realizaciji* teh logičnih vrat in pomnilnih celic – namesto klasičnih tranzistorjev imajo tu osrednjo vlogo kvantne celice.

Vendar pa se ni potrebno omejiti le na realizacije analogne klasičnim. Nekatere strukture se da, kot bo to prikazano na primeru D-celice, prostorsko bolj optimalno realizirati z izkoriščanjem vseh posebnosti kvantnih celic, v tem primeru ure v kvantnih celičnih avtomatih.

4.2 Polni seštevalnik

Kot ponavadi je najprej opisan enostavnejši primer aritmetično-logične sturkture. Gre za t.i. polni seštevalnik (angl. *full adder*), ki je osnova večini nadaljnih postavitev. Zaradi enostavnosti je polni seštevalnik primeren tudi za spoznavanje orodja QCADesigner, predvsem simulacijskega pogona in interpretiranja rezultatov.

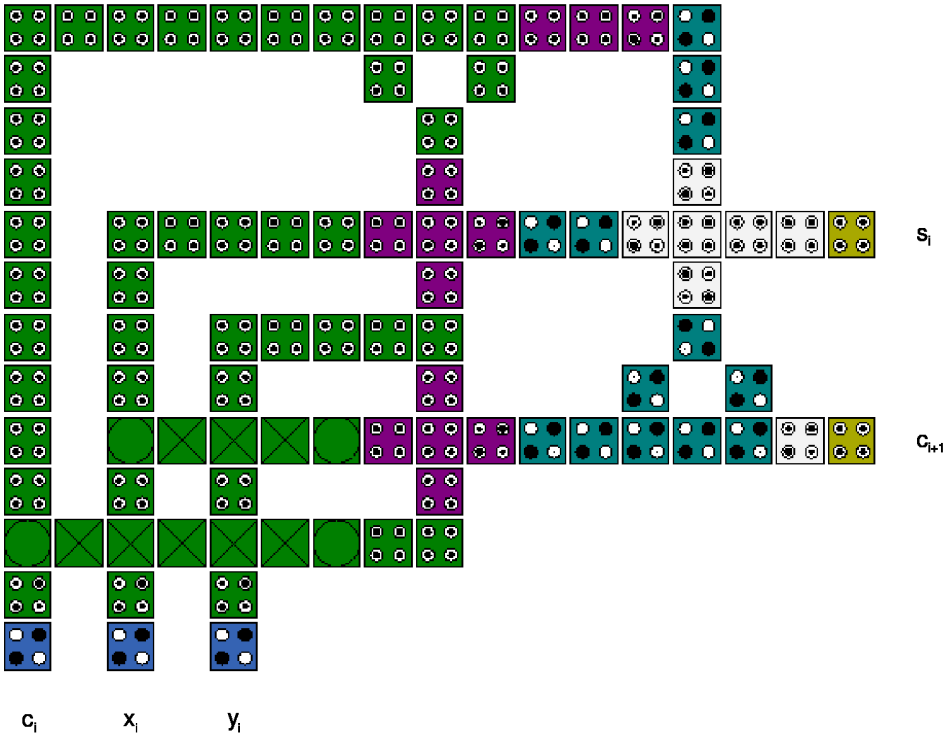
Logično strukturo 1-bitnega polnega seštevalnika in ustrezno pravilnostno tabelo prikazuje slika 4.1.

x_i	y_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Slika 4.1 Logična struktura 1-bitnega seštevalnika in pripadajoča pravilnostna tabela.

4.2.1 Realizacija s kvantnim celičnim avtomatom

Ker gre za enostavnejšo logično strukturo, je mogoče na internetu najti veliko poskusov realizacije le-te s QCADesignerjem. Žal se izkaže, da skoraj nobena ne deluje pravilno za



Slika 4.2 Kvantni celični avtomat, ki realizira 1-bitni seštevalnik. x_i in y_i sta vhoda, na katere pripeljemo števili, ki ju je potrebno sešteti. Vhod c_i služi za povezovanje večih polnih seštevalnikov – nanj je potrebno pripeljati izhod c_{i-1} iz predhodnega polnega seštevalnika. Izhoda sta dva – na s_i se po zakasnitvi ene urine periode pojavi vsota, na c_{i+1} pa prenos.

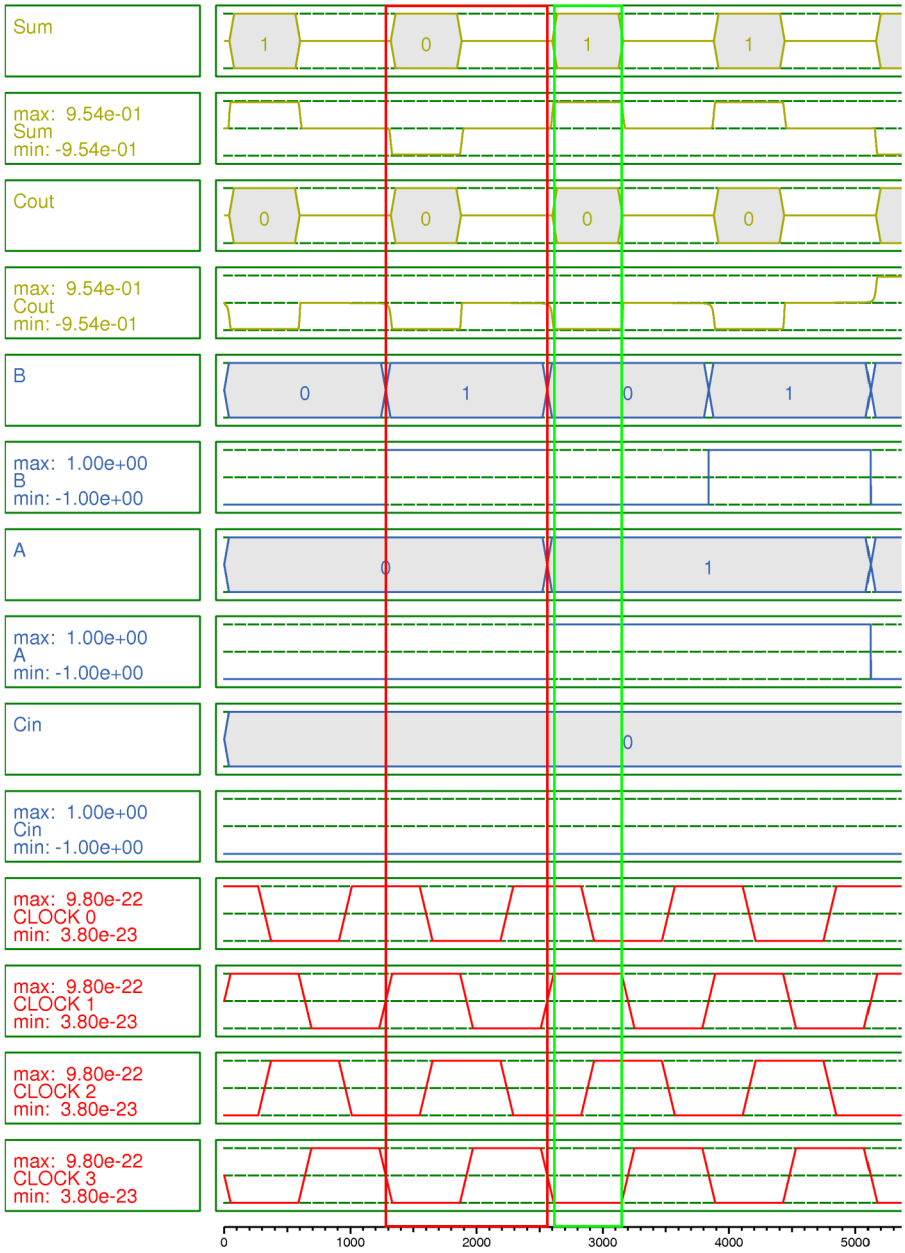
vse možne vhodne vektorje. Edini primer, za katerega lahko trdim, da deluje in sem ga zato tudi uporabljal v kompleksnejših strukturah, je delo idejnega očeta QCADesignerja K. Walusa. Dostopen je na novičarski skupini[1] QCADesignerja, kjer lahko uporabnik izmenjuje izkušnje z ostalimi uporabniki programa in prijavi odkrite napake v orodju.

Realizacijo prikazuje slika 4.2. Uporabljena so tri majoritetna vrata (razložena na sliki 2.11) in dva negatorja (slika 2.10b). Za pravilno delovanje so potrebna še štiri področja ure, ki so na sliki obarvana v skladu z prikazom v QCADesignerju.

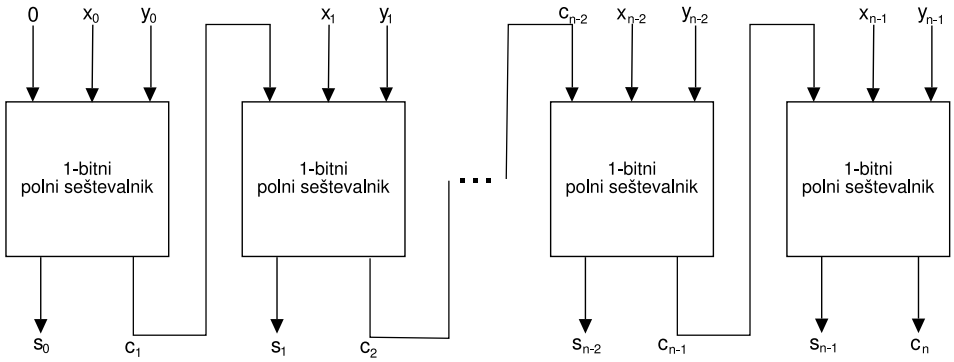
4.2.2 Analiza delovanja

Delovanje strukture za nekaj primerov vhodnih vektorjev prikazuje izpis simulacijskih rezultatov na sliki 4.3. Na sliki so zaradi lažje analize definirana vodila (kljub temu, da

[1] http://tech.groups.yahoo.com/group/qcadesign



Slika 4.3 Izpis rezultatov bistabilne simulacije polnega seštevalnika. Na levem robu so našteti vsi vhodi oz. vodila, levo pa se izrisujejo krivulje signalov. Pri vodilu se najprej izpiše njegova desetiška logična vrednost, pod tem pa so prikazani poteki vseh signalov, ki sestavljajo vodilo. Rdeč okvir označuje čas, ko je na vhodu strukture prisoten vektor $[0, 1, 0]$ oz. $A = 0$, $B = 1$, $C_{in} = 0$, zelen pa čas, ko so na voljo rezultati (izhodni vektor [1,0] oz. $Sum = 1$, $C_{out} = 0$.)



Slika 4.4 Logična struktura n-bitnega seštevalnika s plazovitim prenosom.

je vsako vodilo široko le 1 bit). Prednost vodil je ta, da se rezultati prikazujejo hkrati kot krivulje in kot desetiški zapis logičnih vrednosti, kar precej pohitri pregledovanje.

Vzemimo primer vhodnega vektorja $x_0 = [0]$, $y_0 = [1]$ in $c_0 = [0]$. Spremenljivki x_0 tako ustreza vodilo z oznako A, spremenljivki y_0 vodilo B, prenos c_0 pa je označen z C_{in}. Po podatkih s tabele na sliki 4.1 ustreza vhodnemu vektorju $[0, 1, 0]$ izhod $[1, 0]$, se pravi $s_0 = [1]$ ter $c_1 = [0]$. Spremenljivki s_0 ustreza vodilo Sum, c_1 pa vodilo C_{out}. Rdeč okvir označuje čas v katerem so vhodi stabilni – takrat struktura izračunava izhode. Zelen okvir pa označuje čas, ko so rezultati prisotni na izhodih. Kot je razvidno s slike 4.3, uporablja struktura štiri urina področja, označena s CLOCK $0, 1, 2$ in 3. Rezultati se pojavijo na izhodih takrat, ko preide zadnje urino področje (CLOCK 3) v fazo zadrževanja. V QCADesignerju je to takrat, ko je CLOCK 3 v *nizkem* stanju. V tem času je izhod Sum enak 1, izhod C_{out} pa 0, kar je pravilen rezultat.

4.3 Seštevalnik s plazovitim prenosom

Najpreprostejšo obliko n-bitnega seštevalnika s plazovitim prenosom (angl. *ripple carry adder*) tvori veriga n 1-bitnih seštevalnikov [5]. Logična struktura n-bitnega seštevalnika s plazovitim prenosom je razvidna s slike 4.4. Seštevalnik deluje po naslednjem principu. Na vsak 1-bitni seštevalnik je potrebno pripeljati istoležna bita obeh števil, kar pomeni x_0 in y_0 na prvi seštevalnik, x_1 in y_1 na drugega, in tako dalje do zadnjih dveh bitov x_{n-1} in y_{n-1}. Sprememba signala za *prenos* c_i pa potuje iz najnižjega mesta (najbolj desno na sliki 4.4) proti najvišjemu, kot plaz skozi preostalih $n - 1$ seštevalnikov. Od tod tudi ime seštevalnik s *plazovitim prenosom*. Prvi vhod c_0 na najnižjem mestu mora

biti postavljen na logično 0. Vsota obeh n-bitnih števil je ravno tako dolga n bitov in se pojavi na izhodih s_i.

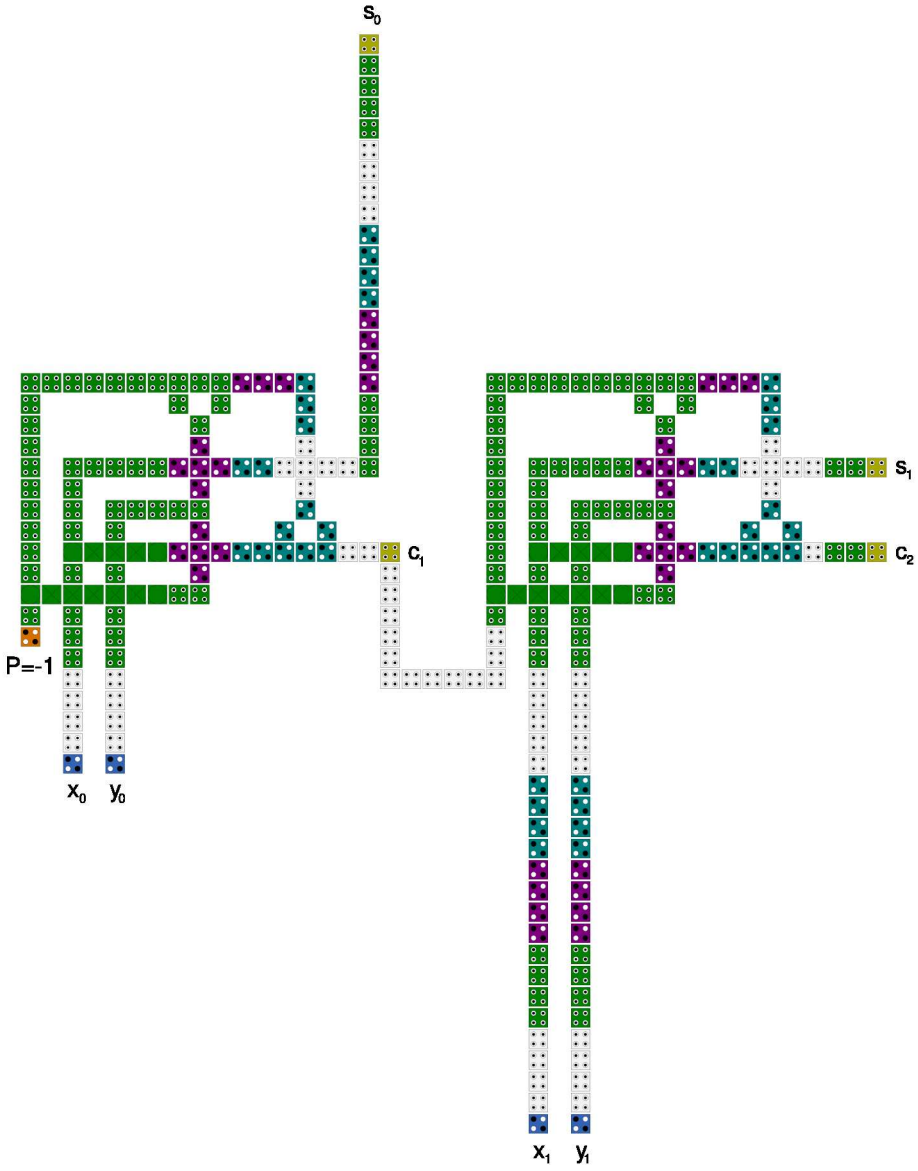
4.3.1 Realizacija s kvantnim celičnim avtomatom

Naslednji korak je realizacija seštevalnika s plazovitim prenosom iz večih 1-bitnih seštevalnikov. Pri vseh aritmetičnih strukturah sem se odločil za realizacijo 2-bitnih struktur, saj s tem nič ne izgubimo na splošnosti. Poljubno večje strukture so namreč samo podvojevanje 2-bitnih – povezovanje poteka po enakem postopku, le število uporabljenih kvantnih celic hitro narašča. Prednost manjših struktur pa je seveda lažje odkrivanje napak in pregledovanje rezultatov ter precej krajši časi simulacij.

V primeru 2-bitnega seštevalnika s plazovitim prenosom je torej potrebno pravilno povezati dva polna seštevalnika. To ni tako preprosta naloga, kot se zdi na prvi pogled, saj je potrebno pri realizaciji s kvantnim celičnim avtomatom upoštevati posebnosti ure, česar konvencionalne logične strukture ne poznajo. Prvi poskus povezovanja po analogiji povezovanja "običajnih" vezij je bil v mojem primeru zato seveda neuspešen. Na srečo so snovalci QCADesignerja v oba simulacijska pogona vgradili zelo uporabno izbiro *Animate*, s pomočjo katere je mogoče spremljati dogajanje v celicah *med* simulacijo. To je zelo uporabno orodje za odkrivanje napak. V mojem primeru se je izkazalo, da so bile povezave med seštevalnikoma pravilne, napačni pa so bili zamiki med fazami ure v različnih delih strukture. Posledično so posamezni deli strukture sicer delovali pravilno, vendar so bili delni rezultati prisotni na povezavah do preostale strukture v napačni urini fazi in se zato niso pravočasno uporabili.

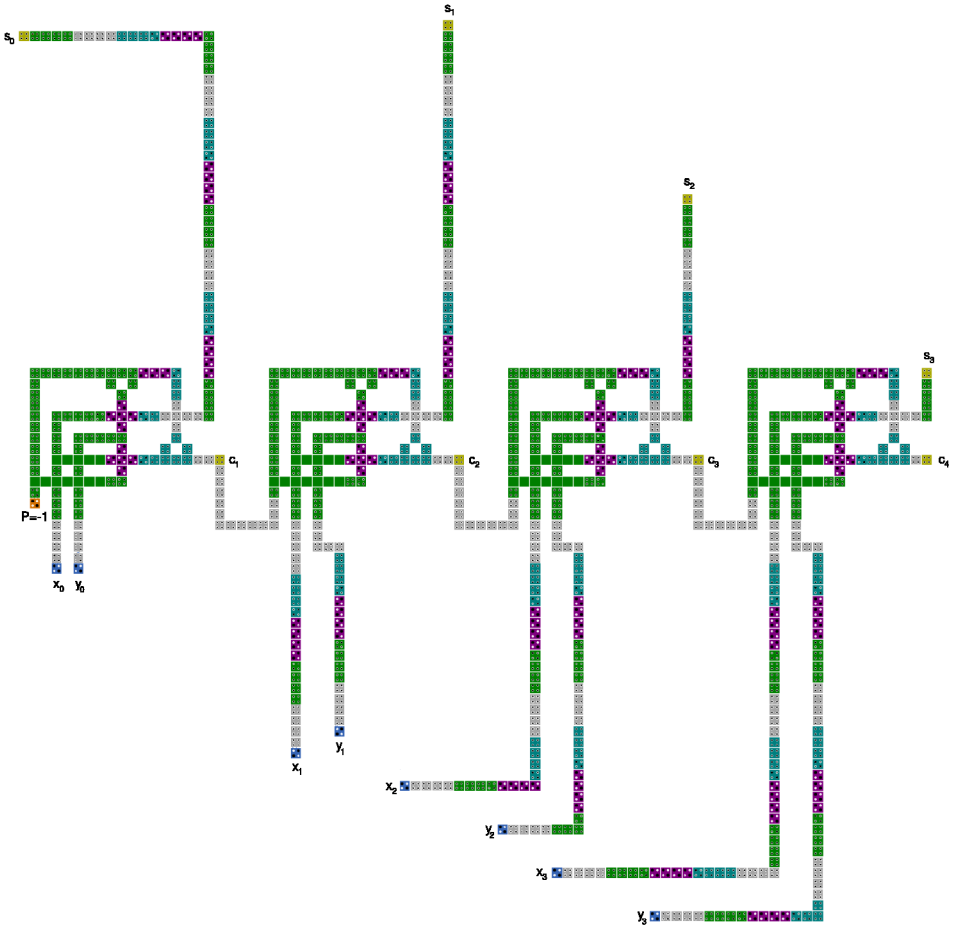
Rešitev je zakasnitev določenih vhodov in izhodov. Pri seštevalniku s plazovitim prenosom se najprej ovrednotijo biti z najmanjšo težo (angl. *least significant bit*), kar v mojem primeru pomeni bita x_0 in y_0 skrajno levo na sliki 4.5. Izračun delnih rezultatov s_0 ter c_1 traja eno urino periodo (vse štiri urine faze). Za enak čas je potrebno zakasniti vhoda x_1 ter y_1, saj je šele sedaj veljaven tudi vhod c_1, ki skupaj s prvima dvema tvori delna rezultata s_1 in c_2 (tudi to traja eno urino periodo). Ko so enkrat vhodi sinhronizirani, je potrebno uskladiti še izhode s_0 in s_1. Prvi je namreč na voljo po eni urini periodi, drugi pa po dveh. Zato je potrebno prvega zakasniti za eno urino periodo. S tem je po dveh urinih periodah glede na stabilne vhode na voljo celoten rezultat $[s_1, s_0]$.

Za ponazoritev kako enostavno je zgraditi večbitne strukture, potem ko je enkrat dodelana 2-bitna različica, je prikazan primer 4-bitnega seštevalnika s plazovitim prenosom.



Slika 4.5 Kvantni celični avtomat, ki realizira seštevalnik s plazovitim prenosom.

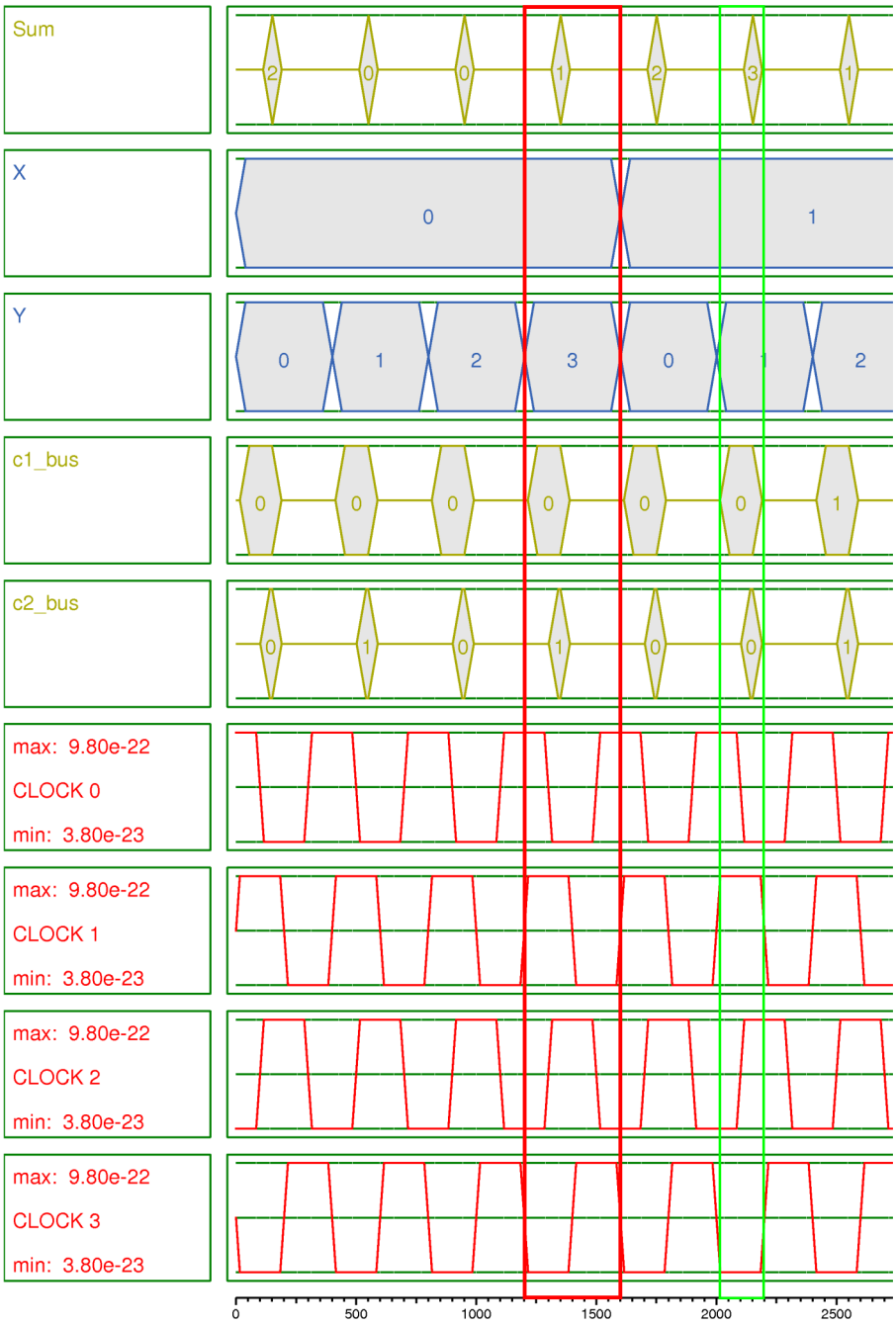
Iz slike 4.6 je razvidno, da velikost strukture, v primerjavi z 2-bitno različico, narašča približno linearno. V tem primeru je za izgradnjo potrebnih 827 QCA celic, temu ustrezno težje pa je tudi iskanje napak in daljši čas simulacije.



Slika 4.6 4-bitni seštevalnik s plazovitim prenosom. Vhodi v posamezne polne seštevalnike morajo biti ustrezno zakasnjeni, da je na razpolago dovolj časa za računanje morebitnih prenosov. Prvi seštevalnik (na sliki skrajno levi) prične z računanjem takoj ko so vhodi stabilni, drugi eno urino periodo za njim, tretji dve urini periodi za prvim, itd. V obratnem vrstnem redu se zakasnijo tudi izhodi iz seštevalnikov (delni rezultati) – izhod iz zadnjega (skrajno desni) nima zakasnitve, predzadnji izhod je zakasnjen za eno urino periodo, predpredzadnji za dve, itd. S tem je zagotovljeno, da se vsi delni rezultati uporabijo hkrati, saj le tako predstavljajo veljaven rezultat.

4.3.2 Analiza delovanja

Slika 4.7 prikazuje delovanje 2-bitnega seštevalnika za nekaj primerov vhodnih vektorjev. Vseh zaradi obsežnosti izpisa ni mogoče prikazati. Iz enakega razloga so prikazana le



Slika 4.7 Izpis rezultatov simulacije 2-bitnega seštevalnika s plazovitim prenosom. V času, označenem z rdečim okvirjem, sta vhoda stabilna in na njiju sta prisotna podatka $X = 0_d = 00_b$ in $Y = 3_d = 11_b$. Po zakasnitvi dveh urinih period (zelen okvir), se rezultat pojavi na izhodih: $Sum = 3_d = 11_b$, prenosa pa ni.

vodila in njihove desetiške vrednosti, ne pa tudi sledi posameznih signalov, ki sestavljajo vodilo (kot je bilo to prikazano pri polnem seštevalniku).

Poglejmo si primer vhodnega vektorja $[X, Y] = [0, 3]$, ki je na sliki 4.7 obrobljen z zelenim okvirjem. Po zakasnitvi dveh urinih period je na voljo rezultat (zelen okvir), t.j. izhodni vektor $[Sum] = [3]$, pri čemer ni prenosa. Podrobneje je dogajanje sledeče. $X = 0_d = 00_b$, $Y = 3_d = 11_b$, kar pomeni da so vhodi v oz. izhodi iz obeh polnih seštevalnikov naslednji:

- prvi seštevalnik: vhod $c_0 = 0$ (fiksno določen), vhod $x_0 = 0$ in vhod $y_0 = 1$; to da izhoda $s_0 = 1$ in $c_1 = 0$ (ni prenosa);
- drugi seštevalnik: vhod $c_1 = 0$ (ker ni bilo prenosa iz prvega seštevalnika), vhod $x_1 = 0$ in vhod $y_1 = 1$; to da izhoda $s_1 = 1$ in $c_2 = 0$ (ni prenosa).

Rezultat je torej $Sum = [s_1, s_0] = [1, 1]$, kar pomeni $11_b = 3_d$.

Za realizacijo preprostega 2-bitnega seštevalnika je v tem primeru potrebnih 311 kvantnih celic. Iz slike je razvidno, da bi bilo za večbitni seštevalnik potrebnih več polnih seštevalnikov ter večje število celic za ustrezno zakasnitev vhodov oz. izhodov. Pri 4-bitnem bi npr. potrebovali 4 polne seštevalnike, najdaljša zakasnitev vhoda oz. izhoda pa bi bila 4 urine periode. Iz tega je mogoče sklepati, da prostorska zahtevnost narašča linearno z dolžino operanda.

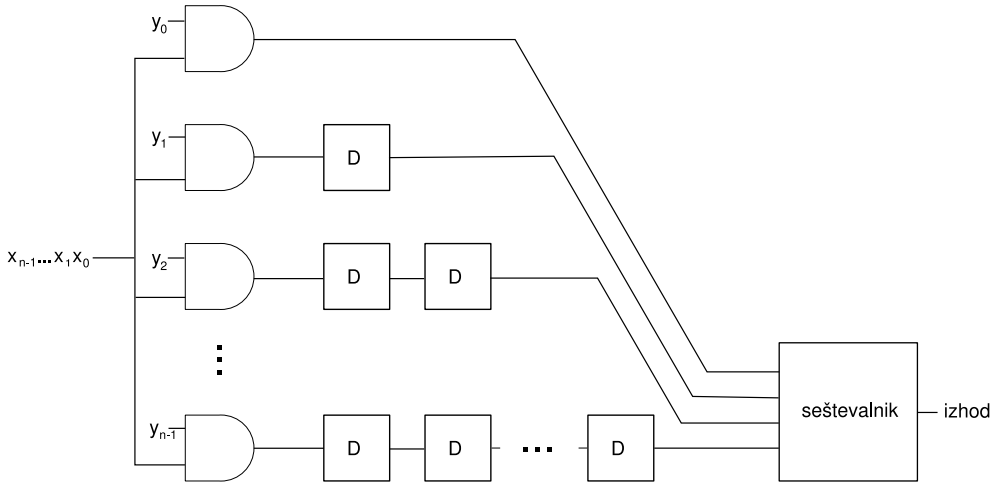
Dobra lastnost take realizacije seštevanja je enostavnost, je pa od vseh možnih realizacij najpočasnejša. Za konvencionalno logično strukturo s tako arhitekturo velja, da trajanje operacije narašča linearno z dolžino operandov (časovna zahtevnost $O(n)$), kar pomeni, da tudi če je posamezen 1-bitni polni seštevalnik hiter, je časovna zakasnitev pri 32- oz. 64-bitnem plazovitem seštevalniku že zelo opazna [5]. Enako velja za kvantno celično strukturo, saj uporablja enako arhitekturo, zakasnitve vhodov/izhodov, potrebne za sinhronizacijo, pa ne vplivajo na celoten čas računanja. Zakasnitev določenih vhodov/izhodov namreč poteka paralelno z računanjem delnih rezultatov.

4.4 Serijsko-paralelni množilnik

Naslednji primer je ena izmed možnih realizacij operacije množenja. Osnova za realizacijo je seštevalnik iz prejšnjega primera. Množenje je sicer mogoče realizirati s čistim kombinatoričnim vezjem, kar pomeni, da se operacija izvede v enem samem koraku. Vendar pa

to zahteva veliko število polnih seštevalnikov, zato se pogosteje uporabljajo enostavnejše realizacije, kot je npr. serijsko-paralelni množilnik (angl. *serial–parallel multiplier*) [6].

Kot rečeno je uporabljen polni seštevalnik iz prejšnjega primera, dodana pa je enostavna kombinatorična logika in D-pomnilna celica (glej sliko 4.8). Ta struktura množi 4-bitna števila, enostavno pa jo je mogoče prilagoditi poljubni dolžini števil. Struktura



Slika 4.8 Logična struktura serijsko-paralelnega množilnika.

deluje podobno kot algoritem za "ročno" množenje – s pomikanjem in seštevanjem. Biti prvega faktorja (x_i) "prihajajo" v vezje serijsko, biti drugega faktorja (y_i) pa paralelno. Biti y_i kontrolirajo AND-vrata na naslednji način. Če je nek bit y_i enak logični 1, se na izhode AND-vrat (vhode seštevalnikov) preko D-celic prenesejo biti x_i, sicer je izhod AND-vrat enak logični 0.

Faktor, ki vstopa v strukutro serijsko, vstopa po vrsti od najmanj pomembnega bita proti najpomembnejšemu (angl. *least significant bit first*). D-pomnilne celice služijo za ustrezno zakasnitev parcialnih produktov, kar ustreza operaciji pomikanja. Nato seštevalnik sešteje vse parcialne produkte v končni rezultat, ki se na izhodu seštevalnika pojavi serijsko.

Delovanje natančneje ponazarja naslednji primer množenja dveh 4-bitnih števil. Dani sta števili 10_d in 14_d, torej 1010_b in 1110_b. Pravilen rezultat je torej $140_d = 10001100_b$. "Ročno" množenje prikazuje spodnja tabela.

				1	**1**	**1**	**0**
			x	**1**	**0**	**1**	**0**
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	**1**	**1**	**1**	**0**	**0**
0	0	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**1**	**1**	**0**	**0**

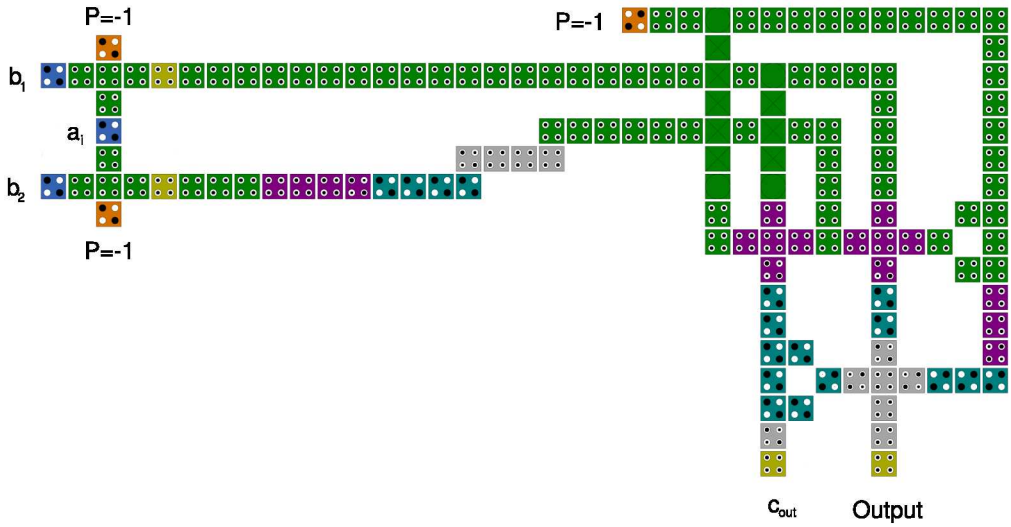
Struktura pa izvede množenje na naslednji način. Prvo število vstopa v strukturo paralelno, torej $y_3 = 1$, $y_2 = 0$, $y_1 = 1$ ter $y_0 = 0$, drugo število pa serijsko, kar pomeni $x_3 = 1$, $x_2 = 1$, $x_1 = 1$ ter $x_0 = 0$. Potek množenja in posamezni parcialni produkti so prikazani v naslednji tabeli.

	P7	P6	P5	P4	P3	P2	P1	P0
A	0	0	0	0	**0**	**0**	**0**	**0**
B	0	0	0	**1**	**1**	**1**	**0**	**0**
C	0	0	**0**	**0**	**0**	**0**	**0**	**0**
D	0	**1**	**1**	**1**	**0**	**0**	**0**	**0**
izhod	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**

Kot je razvidno iz tabele so parcialni produkti na vhodu A v seštevalnik vedno enaki 0, saj je v izbranem primeru bit y_0 enak 0. Podobno je tudi z vhodom C, katerega krmili y_2. Vhod B je enak faktorju 1110_b, zakasnjenem za eno enoto (za eno urino periodo). Vhod D pa je prav tako enak faktorju 1110_b, vendar zakasnjen za tri enote. Seštevalnik sešteje vse te parcialne produkte in na izhodu se biti rezultata 10001100_b pojavijo serijsko. Najprej se pojavi najmanj pomemben bit, nato si sledijo ostali do najpomembnejšega. Če sta faktorja dolžine n, je rezultat dolžine $2n$ – v tem primeru je torej rezultat dolg 8 bitov.

4.4.1 Realizacija s kvantnim celičnim avtomatom

Kvantni celični avtomat, ki realizira serijsko-paralelni množilnik, je prikazan na sliki 4.9. AND-vrata na vhodih so enaka kot na sliki 2.12. Polni seštevalnik, ki sešteva delne rezultate, je enak kot v prejšnjih primerih, D-pomnilna celica, ki služi za zakasnitev parcialnih produktov, pa je drugačna kot v podpoglavju 2.4.1. Zakasnitev za eno urino periodo se namreč enostavneje doseže z nizom celic, ki je ustrezno razdeljen na urina



Slika 4.9 Kvantni celični avtomat, ki realizira serijsko-paralelni množilnik.

področja. Taka realizacija (na sliki je vidna desno od vhoda b_2) zahteva manj kvantnih celic kot rešitev predstavljena v podpoglavju 2.4.1.

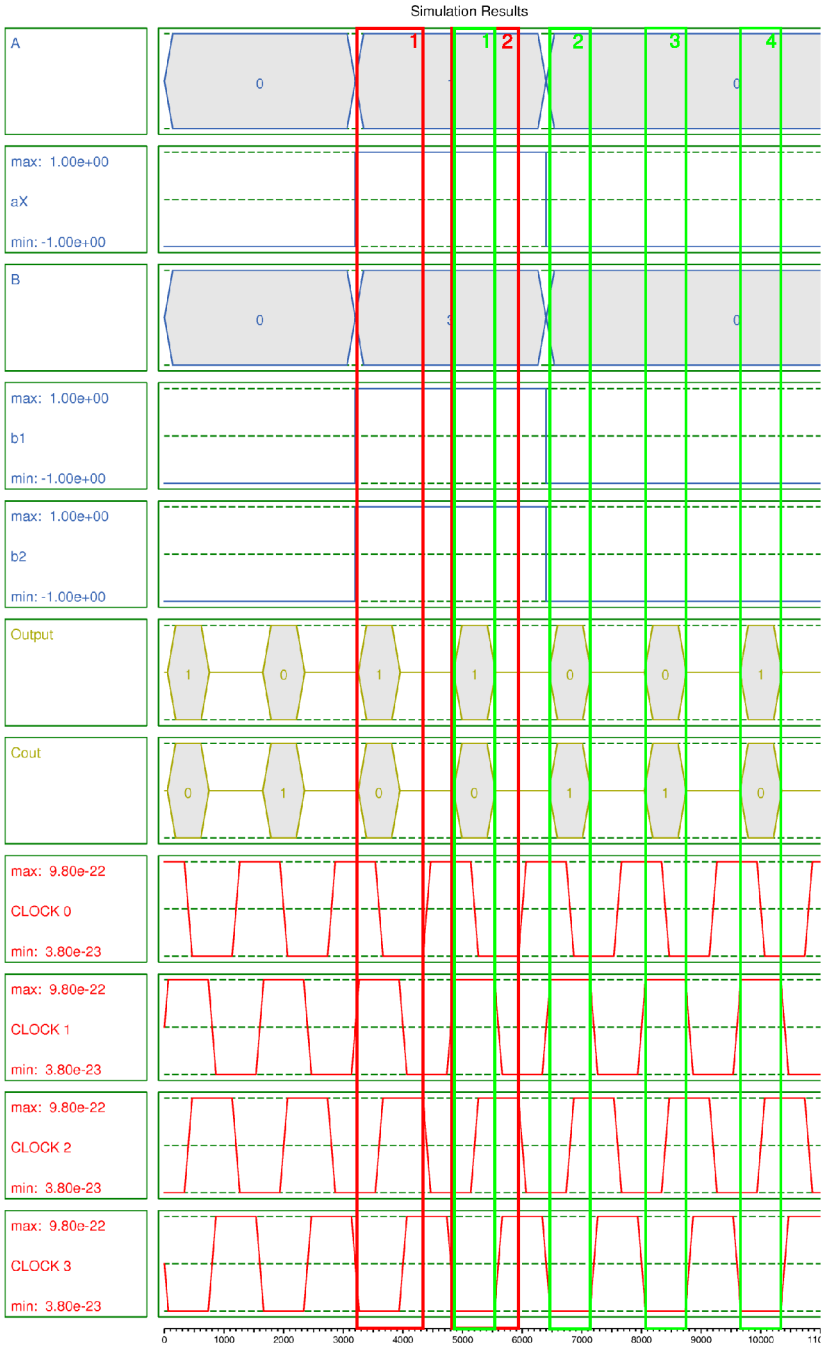
4.4.2 Analiza delovanja

Posebnost pri tej strukturi je to, da en operand vstopa vanjo serijsko (vhod a_i), drugi pa paralelno (vhoda b_1 in b_2). Za realizacijo tega, bi bil na vhodu a_i potreben pomikalni register, ki bi posamezne bite serijskega operanda "spuščal" v strukturo. Realizaciji registra sem se izognil z prilagoditvijo vhodnih vektorjev. Primer množenja vektorja $A = [0, 1]$ in $B = [1, 0]$ prikazuje naslednja tabela.

a_i	b_1	b_2
0	1	0
1	1	0

Vektor A prihaja serijsko – v prvi urini periodi pride bit 0, v naslednji pa bit 1. Med tem časom mora biti na vhodih v strukturo ves čas prisoten vektor B, zato se v tabeli dvakrat ponovi.

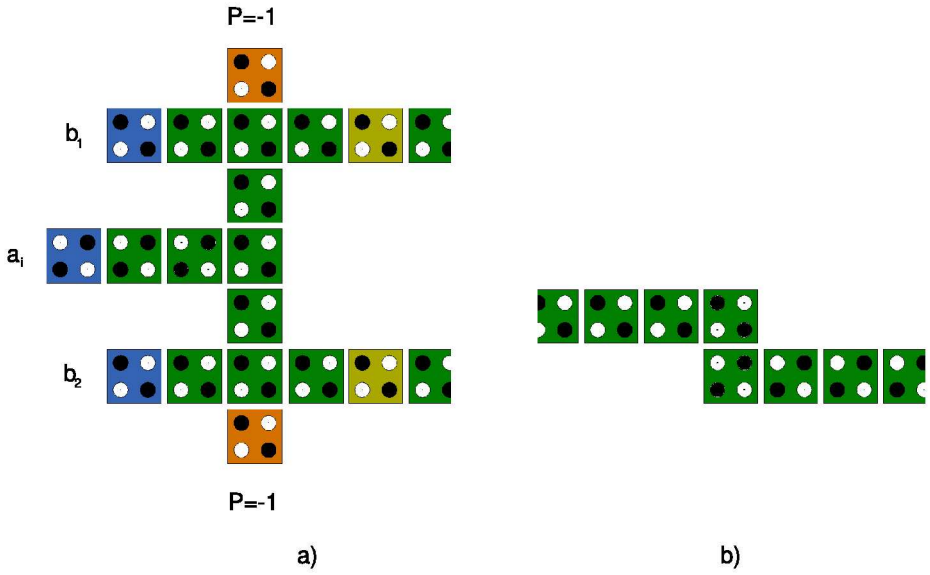
Slika 4.10 prikazuje izpis rezultatov simulacije za množenje vektorjev $A = [1, 1]$ in $B = [1, 1]$. Ustrezna tabela vhodnih vektorjev za simulacijski pogon je naslednja.



Slika 4.10 Izpis simulacije 2-bitnega serijsko-paralelnega množilnika.

a_i	b_1	b_2
1	1	1
1	1	1

Pravilen rezultat je torej $[1, 0, 0, 1]$, pri čemer se prenos pojavi pri obeh ničlah. Na sliki se v času, ki ga označuje rdeč okvir s številko 1, na vhodih strukture pojavita vektorja $A = [1]$ (*prvi* bit vektorja A) in $B = [1, 1]$. Rezultata teh vhodov sta dva, [1] ter [0], in sta prikazana v zelenih okvirjih 1 in 2. Naslednji vhodi so ponovno $A = [1]$ (*drugi* bit vektorja A) in $B = [1, 1]$. Rezultata sta [0] ter [1], kar prikazujeta zelena okvirja 3 in 4. Če serijo delnih rezultatov sestavimo, dobimo pravilen rezultat $Output = [1, 0, 0, 1]$. Če preverimo še prenos, vidimo da se $Cout$ pravilno postavi takrat, ko sta na izhodu $Output$ obe ničli.



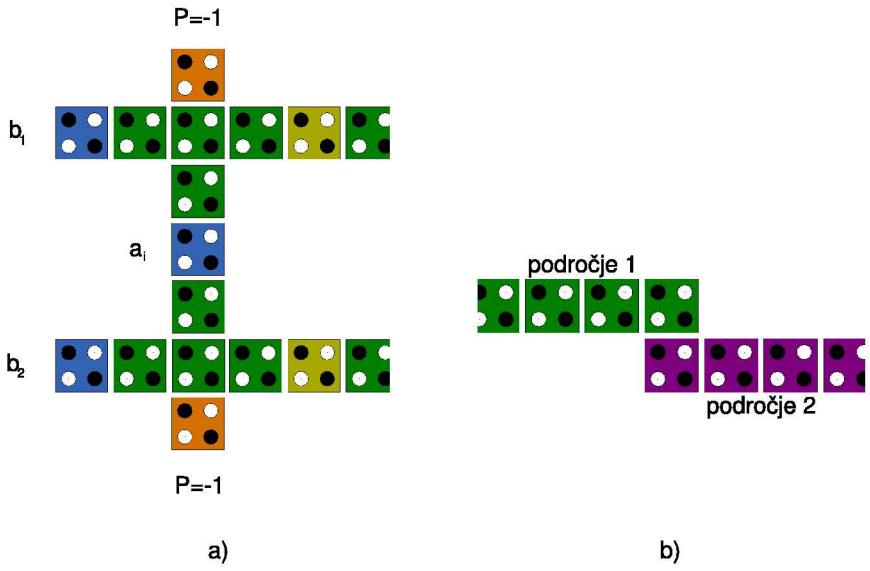
Slika 4.11 Napake pri simulaciji: a) nepravilno razmnoževanje signala a_i ter b) nepravilen prenos po liniji (enaka napaka se pojavi, če sta liniji zamaknjeni, ali pa če imamo pravokoten "ovinek".

Vse pa le ni tako idealno kot zgleda iz primera. Simulacija namreč pri nekaterih vhodih ne daje pravilnih rezultatov. Po podrobnejšem pregledu delovanja strukture s pomočjo izbire *Animate* sem našel dva pojava, ki jih nisem pričakoval (glej sliko 4.11).

Problem je v tem, da se stanje celice v nekaterih primerih ne prenese pravilno preko "vogalov" linij. V mojem primeru se je to dogajalo pri razmnoževanju vhoda A (slika 4.11a), ki ga je potrebno pripeljati na oboja AND-vrata ter pri pravokotnem lomljenju

vhodne linije (slika 4.11b) v polni seštevalnik.

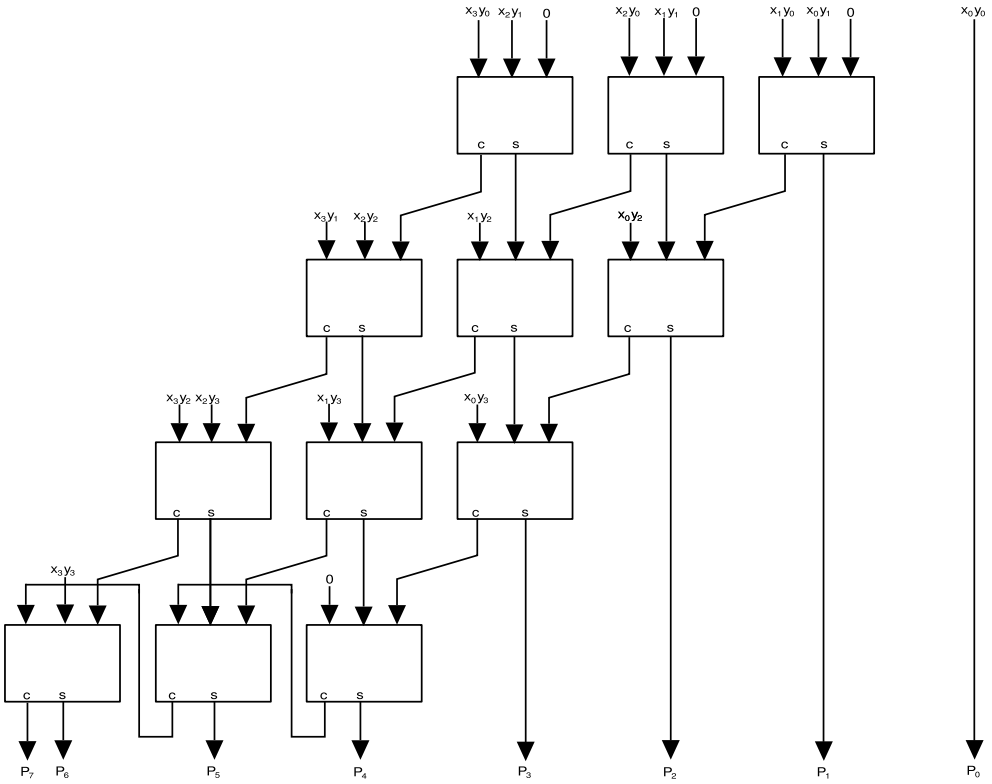
Razmnoževanje vhoda sem popravil tako, da sem vhodno celico postavil "znotraj" linije, tako da signala izhajata neposredno iz nje (glej sliko 4.12a). Napake pri pravokotnem lomljenju linije pa ni mogoče popraviti zgolj z drugačno razporeditvijo celic. Edina rešitev, ki se je obnesla v mojem primeru, je uporaba dveh urinih področji (slika 4.12b), zaradi česar pa se daljša čas računanja.



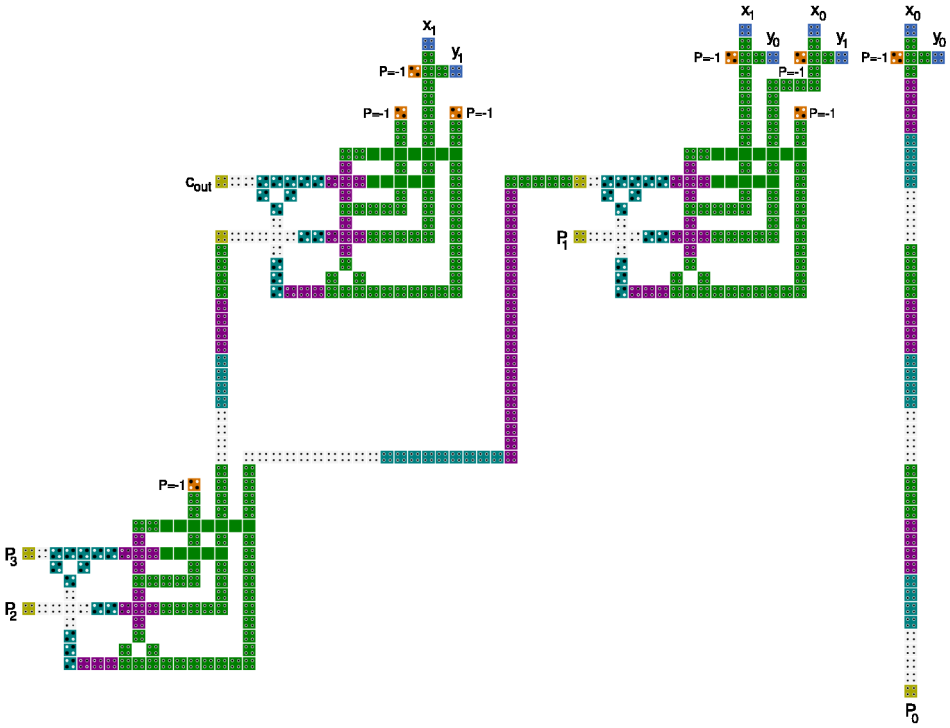
Slika 4.12 Rešitev napak s slike 4.11: a) odpravljanje napačnega razmnoževanja signala ter b) pravilno lomljenje linije.

4.5 Matrični množilnik

Matrični množilnik (angl. *array multiplier*) ali tudi kombinatorični množilnik (angl. *combinatorial multiplier*) je čisto kombinatorično vezje, ki izvaja operacijo množenja s pomočjo seštevanja ustrezno pomaknjenih števil. Princip delovanja je podoben kot v prejšnjem primeru, le da je delovanje precej hitrejše, saj za pomike ni uporabljena pomnilna celica. Cena za tako hitrost pa je, kot je to običajno pri čistih kombinatoričnih strukturah, zahteva po velikem številu logičnih elementov in posledično višja cena. Za realizacijo množenja dveh n-bitnih števil je potrebnih $(n-1)^2$ 1-bitnih polnih seštevalnikov ter n^2 AND-vrat [5]. Logično strukturo takega množilnika prikazuje slika 4.13, iz katere je razvidno, da je časovna kompleksnost take strukture $O(n)$. To je na prvi pogled zelo počasno,



Slika 4.13 Logična struktura matričnega množilnika za množenje dveh 4-bitnih števil.

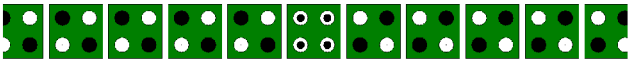


Slika 4.14 Kvantni celični avtomat, ki realizira matrični množilnik.

posebej za kombinatorično vezje. Vendar to ni tak problem, saj struktura omogoča *cevovodno procesiranje*. Matrika seštevalnikov ima namreč n vrstic, pri čemer je posamezna vrstica pri enem množenju zasedena le $1/n$-ti del časa. To pomeni, da je mogoče pričeti z naslednjim množenjem takoj, ko je prva vrstica matrike prosta in ni potrebno čakati do konca predhodne operacije. Na ta način je mogoče z matričnim množilnikom dobiti (po začetni zakasnitvi) produkt v času, ki ni odvisen od n.

4.5.1 Realizacija s kvantnim celičnim avtomatom

Avtomat, ki realizira množenje dveh 2-bitnih števil, prikazuje slika 4.14. Struktura je že kar obsežna (413 celic), saj se želimo izogniti upočasnitvi, ki nastane zaradi uporabe pomnilnih celic (prejšnji primeri). V tem primeru so potrebni trije polni seštevalniki in štiri AND-vrata. Naloga polnih seštevalnikov je, kot v prejšnjem primeru, seštevanje delnih produktov v končni rezultat. AND-vrata pa služijo združevanju vseh možnih kombinacij vhodnih bitov, kar je potrebno storiti preden le-ti pridejo do prvih polnih



Slika 4.15 Napaka na liniji.

seštevalnikov.

4.5.2 Analiza delovanja

Po pričakovanjih je s to strukturo največ težav pri preverjanju delovanja. Zaradi velikosti je tudi uporaba funkcije *Animate* težavnejša, saj je nemogoče spremljati spremembe stanj vseh celic znotraj istega urinega področja. Zato je edina rešitev preverjanje strukture po delih. To pa pomeni veliko ponovitev simulacije pri istih vhodnih vekorjih, s čimer se izgubi precej časa. Dodatna težava je še v tem, da simulacija pri računanju vplivov ene celice na drugo naključno izbira vrstni red celic. To je s stališča čimbolj realne simulacije sicer pravilno, vendar pa zaradi tega rezultati niso vedno povsem enaki, kar dodatno otežuje delo.

Podobno kot v prejšnjem primeru, sem tudi tu zasledil pojave, ki so povzročali nepravilno delovanje strukture. Napake pri pravokotnem lomljenju linije so tu precej bolj pereče, saj se jim je zaradi velikosti strukture nemogoče povsem izogniti samo s prerazporeditvijo strukture in uporabo samih ravnih linij. Potrebno je vpeljati nova urina področja.

Še bolj pereč pa je naslednji problem, za katerega pa žal nisem našel uporabne rešitve. Na linijah, ki povezujejo AND-vrata z vhodom v polni seštevalnik, se stanja celic včasih negirajo kot to prikazuje slika 4.15. Celica z napačnim stanjem (nepolarizirana) je s stališča simulacije povsem enaka kot ostale celice, zato brisanje linije in ponovno postavljanje ne pomaga. Do napake tudi ne pride vedno na istem mestu (isti celici), ampak se včasih pojavi npr. deset celic stran od prejšnje pojavitve. Izključil sem tudi morebitne vplive sosednjih linij (presluh), saj se napaka pojavlja tudi na edini liniji med dvema deloma strukture. Tako obnašanje celic lahko pripišem edino načinu simuliranja dogajanja v strukturi (mogoče na to vpliva naključni vrstni red izbiranja celic), ali pa fizikalnim pojavom, ki pa presegajo okvire tega diplomskega dela in se vanje nisem poglabljal.

Zaradi omenjenih težav, pri tej strukturi ni izpisa rezultatov simulacije, saj so razultati napačni. Žal se s QCADesignerjem ne da izpisati sledi signala na npr. eni liniji

strukture ali pa stanj vseh celic, ki sestavljajo linijo, da bi lahko analiziral dogajanje v posameznih celicah v trenutku pojavitve napake in tako natančneje ugotovil problematične okoliščine.

5 Zaključek

Nanotehnologija je zadnji dve desetletji vsekakor ena najbolj obetajočih področij znanosti in je že davno prerasla okvire fizikalnih laboratorijev. To pa pomeni tudi to, da se z njo ukvarja vedno več znanstvenikov, ki jih podrobnosti fizikalnih procesov ne zanimajo in želijo čimbolj raziskati praktično uporabo teh izsledkov. QCADesigner je po mojem mnenju dobra ideja za premostitev tega prepada.

Pri realizaciji aritmetično-logičnih struktur, sem se z orodjem QCADesigner dodobra spoznal. Izkazalo se je, da v smislu olajšanja spoznavanja z novim področjem znanosti dobro služi svojemu namenu. Boljše razlage kvantnih celičnih avtomatov, kot te, da dobesedno opazujemo dogajanje v sami strukturi med delovanjem, si skorajda ne predstavljam. To zelo olajša razumevanje pojmov, kot je npr. ura v kvantnih celičnih avtomatih. Ta ima precej drugačno vlogo, kot smo je vajeni iz konvencionalnih struktur.

Pri delu z nekoliko zahtevnejšimi strukturami se sicer pokažejo tudi nekatere nedodelanosti in pomankljivosti. Simulacije osnovnih celičnih struktur, kot so npr. majoritetna vrata, multiplekser, negator, polni seštevalnik, delujejo brez težav, kar je za začetnika razveseljivo. Težave se pojavijo, ko struktura po obsegu naraste in vsebuje več med seboj

povezanih logičnih celot. Pri opazovanju dogajanja med simulacijo, se je težko otresti vtisa, da deluje vsaka logična celota nekoliko "po svoje", na linijah, ki jih povezujejo, pa se nato dogajajo "čudni" pojavi. Eden takih primerov je negiranje stanja celic sredi linije pri matričnem množilniku. Na vprašanje ali imajo te napake fizikalno ozadje v kaotičnih sistemih, ali pa so stranski produkt algoritma simulacije, ne znam odgovoriti. Taka ocena presega moje znanje in obseg tega diplomskega dela.

Četudi orodju odpustimo razne nedodelanosti in občasno nestabilnost, pa ima eno očitno pomankljivost. Podpore za iskanje napak v delovanju strukture praktično ni. Do neke mere jo pri manjših strukturah nadomešča izbira *Animate*, s pomočjo katere lahko sledimo (ne)delovanju. Kakor hitro pa struktura preseže mejo npr. 100 kvantnih celic, postane s pogledom zelo težko slediti vsem spremembam. Tu pogrešam orodja, ki so že dolgo železni repertoar okolij za izdelavo programske opreme. Dobrodošla bi bila možnost izpisovanja stanja izbranih celic med animirano simulacijo. S tem bi lahko natančneje opazovali interakcijo med celicami, ki se obnašajo proti našim pričakovanjem. Prav tako bi bila zelo dobrodošla možnost izvajanja simulacije po korakih ali pa vsaj prekinjanje simulacije in kasnejše nadaljevanje od prekinitve naprej.

Ko bi te osnovne funkcije delovale, bi lahko simulator dopolnili tako, da bi npr. upošteval energijo, ki prihaja v strukturo iz okolice. Lahko bi dodali tudi podporo celicam z več kot le dvema stabilnima stanjema, itd.

Simulator pa že sedaj podpira nekatere funkcije, za katere se mi zdi, da niso dovolj izkoriščene s strani uporabnikov. Taka je npr. podpora večplastnim strukturam. Zaenkrat smo uporabniki to uporabljali le za premoščanje linij, nikjer pa nisem zasledil, da bi kdo skušal analizirati in izkoristiti medsebojne vplive celic, ki se nahajajo na različnih plasteh – torej razširitev enoplastnih struktur v večplastna.

Prav tako je precej očitno, da je večina kvantnih celičnih struktur narejena po analogiji s strukturami iz konvencionalnih vezij. Tudi moji primeri niso nobena izjema. Mogoče bi bilo dobro poskusiti razviti strukture, ki bi ob enaki funkciji bolje izkoriščale vse posebnosti kvantnih celic. Lep primer tega je D-pomnilna celica, ki je v različici, narejeni po analogiji iz konvencionalnih vezji, precej obsežnejša, kot pa različica, ki izkorišča kar urina področja.

To je le nekaj vprašanj, ki se pojavljajo ob delu z novo tehnologijo, in bodo morala dobiti zadovoljive odgovore preden bomo sadove nanotehnologije izkoriščali tudi v praksi.

LITERATURA

[1] C. Lent, P. Tougaw, W. Porod, G. Bernstein, Quantum cellular automata, Nanotechnology 4 (1993) 49–57.

[2] M. T. Niemier, P. M. Kogge, Origins and motivations for design rules in QCA, in: S. Shukla, R. Bahar (Eds.), Nano, quantum and molecular computing, Kluwer Academic Publishers, Boston, 2004, pp. 267–293.

[3] C. S. Lent, The concept of quantum-dot cellular automata, in: M. Macucci (Ed.), Quantum cellular automata, Imperial College Press, London, 2006, pp. 1–14.

[4] P. Pečar, Uporaba adiabatnega pristopa pri realizaciji trojiškega procesiranja na osnovi kvantnih celičnih avtomatov, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Ljubljana, 2007.

[5] D. Kodek, Arhitektura računalniških sistemov, Bi-Tim, Ljubljana, 2000.

[6] A. Almaini, Electronic logic systems, Prentice-Hall International, London, 1986.

Izjavljam, da sem diplomsko nalogo izdelal samostojno. Vse, ki so mi pri tem pomagali, sem navedel v zahvali.

— Tomaž Orač, Ljubljana, september 2007.