Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Janež

Metode razmeščanja in povezovanja logičnih primitivov kvantnih celičnih avtomatov

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Miha Mraz

Ljubljana, 2012

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Janež

Metode razmeščanja in povezovanja logičnih primitivov kvantnih celičnih avtomatov

POVZETEK

Zaradi nenehnega razvoja se miniaturizacija integriranih vezij, realiziranih v CMOS tehnologiji, približuje svojim mejam. Zato se je pojavilo več predlogov novih tehnologij, ki bi dejavnike za omejitve CMOS tehnologije izkoristile sebi v prid. Eden izmed predlogov so naprave na osnovi kvantnih celičnih avtomatov (QCA). Njihov osnovni gradnik je površinska celica s štirimi kvantnimi pikami in z dvema elektronoma, ki se lahko znotraj celice razporedita v dve legi. Elektrostatični vplivi omogočajo prenos podatkov med celicami. S tem so kvantni celični avtomati primerni za dvovrednostno procesiranje. Raziskovalci so iz celic sestavili logične primitive, kot so majoritetna vrata, negator in prenosna linija. Tako so postavili temelje za gradnjo kompleksnih struktur QCA. Za njihovo snovanje je tako kot v primeru CMOS tehnologije nujno formalizirati metodologijo snovanja in vpeljati računalniško podporo.

V disertaciji obravnavamo avtomatizacijo snovanja fizične razmestitve strukture kvantnih celičnih avtomatov. Snovanje je sestavljeno iz stopenj razmeščanja in povezovanja logičnih primitivov. Razvili smo metode za avtomatsko razmeščanje in povezovanje logičnih primitivov QCA in tako vpeljali računalniško podporo na področje snovanja struktur QCA. Analizirali smo obstoječe algoritme za snovanje integriranih vezij v CMOS tehnologiji in jih priredili za snovanje struktur QCA. Za razmeščanje logičnih primitivov QCA smo uporabili algoritem simuliranega ohlajanja. Za povezovanje razmeščenih primitvov smo razvili algoritem, ki temelji na iskanju po labirintu in iskanju po liniji.

Pravilno delujoče strukture je možno zasnovati le z uporabo načrtovalskih pravil, ki določajo potrebne lastnosti strukture. Načrtovalska pravila morajo zato biti upoštevana tudi v računalniškem orodju za avtomatsko snovanje struktur QCA. V disertaciji smo

definirali načrtovalska pravila, ki jim mora zadoščati pravilno delujoča struktura. Definirana načrtovalska pravila upoštevajo fizikalne značilnosti QCA, zato lahko z njegovo uporabo zasnujemo tehnološko izvedljive strukture.

Za potrebe ovrednotenja lastnosti avtomatsko zasnovanih struktur QCA smo določili ustrezne metrike. Na njihovi osnovi smo ocenili kvaliteto avtomatsko zasnovanih struktur. Na podlagi vpeljanih metrik smo tudi primerjali avtomatsko zasnovane in obstoječe ročno zasnovane strukture QCA.

Implementirali smo računalniško orodje za avtomatsko snovanje struktur QCA. Z njegovo uporabo smo zasnovali različne strukture QCA in analizirali rezultate snovanja. Pravilno delovanje vseh zasnovanih struktur smo verificirali s simulacijo v orodju QCADesigner.

Ključne besede: kvantni celični avtomati, nekonvencionalno procesiranje, računalniško podprto snovanje, razmeščanje, povezovanje, načrtovalska pravila

University of Ljubljana
Faculty of Computer and Information Science

Miha Janež

Methods for placement and routing of quantum-dot cellular automata logic gates

ABSTRACT

Due to continuous development, the miniaturization of the integrated circuits in CMOS technology is approaching its limit. Thus emerged several proposals for new technologies that would take advantage of the effects which hinder the operation of miniaturized CMOS circuits. One of the proposed novel technologies is the quantum-dot cellular automaton (QCA). Its basic building block is a QCA cell with four quantum dots and two electrons which can assume two configurations within a cell. Electrostatic effects enable the transfer of data between cells. Therefore, the QCA can perform binary processing. Researchers constructed various logic gates composed of QCA cells, including the majority gate, the inverter and the wire. This enables the construction of complex QCA structures. As in the design of CMOS circuits, the QCA design methodology must be formalized and computer aided design must be introduced.

In this thesis we address the automatization of the layout design of QCA structures. Layout design consists of placement and routing of logic gates. We developed the methods for automatic placement and routing of QCA logic gates, thus introducing the computer aided design of QCA structures. We analyzed existing algorithms for layout design in CMOS technology and adapted them for layout design of QCA structures. For placement of QCA logic gates we used the simulated annealing algorithm. For routing of placed gates we developed the algorithm based on the maze router and the line probe search.

Correctly operating structures can be designed only by using the design rules that determine the necessary structure characteristics. Thus, the design rules must be considered by the computer tool for automatic QCA layout design. In thesis we defined the design rules that must be satisfied by correctly operating structure. The defined design

rules take into account the physical characteristics of QCA, thus their use enables the design of manufacturable structures.

In order to evaluate the characteristics of automatically designed QCA structures we defined appropriate metrics. They were used to assess the quality of automatically designed structures. Based on the introduced metrics we compared automatically designed and manually designed QCA structures.

We implemented the computer tool for automatic layout design of QCA structures. We used the tool to design several QCA structures and we analyzed the results. Correct operation of all designed structures was verified by the simulation in QCADesigner tool.

Keywords: quantum-dot cellular automata, unconventional computing, computer aided design, placement, routing, design rules

ZAHVALA

Zahvaljujem se vsem, ki so kakorkoli pomagali pri izdelavi pričujočega dela, še posebej mentorju prof. dr. Mihi Mrazu in sodelavcem v Laboratoriju za računalniške strukture in sisteme.

— Miha Janež, Ljubljana, april 2012

KAZALO

1 Uvod

1.1 Uvod

Danes delovanje večine digitalnih elektronskih naprav omogočajo integrirana vezja, realizirana na osnovi CMOS tehnologije. Zaradi nenehnega razvoja in potreb po učinkovitejših napravah se vezja stalno manjšajo. Po Moorovem zakonu se število tranzistorjev (osnovnih gradnikov vezja) na enaki površini vsaki dve leti podvoji [1]. Z miniaturizacijo se veča tudi hitrost delovanja vezja, vendar se s približevanjem tranzistorjev nanometrskim velikostim pojavljajo tudi težave, pogojene s kvantnimi efekti [2]. Za premostitev teh težav so se začele porajati ideje o napravah, ki jih zakoni kvantne fizike ne bi omejevali, temveč bi jih naprave celo izkoriščale. Kot ena izmed možnih alternativ CMOS tehnologiji so se pojavili *kvantni celični avtomati* (angl. *Quantum-dot Cellular Automata - QCA*) [3]. Ta tehnologija za procesiranje izkorišča kvantno tuneliranje, ki postavlja mejo za miniaturizacijo klasičnih tranzistorjev. Na tej osnovi naj bi nove naprave porabile manj energije, pri tem pa bi bile manjše in hitrejše od današnjih.

Osnovni gradnik kvantnega celičnega avtomata je vpeljal C. S. Lent leta 1993. Gre za površinsko celico z dvema elektronoma in s štirimi kvantnimi pikami, razporejenimi

v oglišča kvadrata. Posamezen elektron se nahaja v eni izmed pik. Zaradi delovanja Coulombove sile se elektrona v izolirani celici razporedita v eno od dveh diagonalnih leg. Slednji sovpadata s pomenom dvovrednostnega logičnega stanje celice. Procesiranje in prenos podatkov med celicami v avtomatu omogočajo medsebojni elektrostatični vplivi. Manjše število celic sestavlja logični primitiv, ki izvaja določeno logično operacijo, podobno kot logična vrata implementirana s CMOS tehnologijo. Najpomembnejši predstavljeni primitivi so *majoritetna vrata*, *negator* in *prenosna linija* (angl. *QCA wire*) [4]. Z majoritetnimi vrati je možno realizirati logični operaciji konjunkcije in disjunkcije, tako da našteti logični primitivi tvorijo funkcijsko poln nabor dvojiških logičnih funkcij. Za realizacijo procesno uporabnega sistema je potrebno sestaviti kompleksno strukturo iz velikega števila logičnih primitivov, ki so medsebojno povezani z linijami. Manjše strukture, kot sta seštevalnik in množilnik [4, 5, 6], lahko zasnujemo ročno, za snovanje večjih pa je nujno potrebno vpeljati metodologijo snovanja in ustrezno avtomatizirano podporo.

Izum polprevodniških tranzistorjev sredi dvajsetega stoletja je omogočil izdelavo elektronskih integriranih vezij. V prvih letih njihove izdelave so posamezni proizvajalci razvili lastne načine snovanja. Z manjšanjem tranzistorjev, večanjem njihove gostote na čipu in posledično kompleksnejšimi vezji, se je pojavila potreba po formalizaciji enotne metodologije snovanja. Slednjo sta skupaj z načrtovalskimi pravili vpeljala avtorja Mead in Conway [7]. To je vzpostavilo skupen jezik med snovalci sistemov in proizvodnimi strokovnjaki, kar je omogočilo razvoj algoritmov za snovanje obsežnejših integriranih vezij. Z uporabo metodologije in ob upoštevanju načrtovalskih pravil lahko sistemski arhitekti zasnujejo kompleksno vezje, ki bo zadoščalo danim zahtevam in ga bo možno izdelati. Pod pojmom snovanja ločujemo med logičnim in fizičnim segmentom [8, 9, 10, 11]. Pri prvem ustvarimo logično shemo vezja, rezultat fizičnega segmenta pa je fizična razmestitev realizacije vezja. Pričujoče delo obravnava fizično snovanje, katerega osnovni cilj je ustrezna razmestitev in povezava gradnikov vezja, ki realizirajo entitete v logični shemi.

Hkrati z razvojem tehnologije za izdelavo integriranih vezij so se razvijale in nadgrajevale tudi metode njihovega snovanja. Cilj metod *razmeščanja* (angl. *placement*) je določiti položaje gradnikov na čipu, tako da bo vezje možno fizično realizirati, pri tem pa se optimizirajo različni kriteriji. Tipični so skupna dolžina povezav, dolžina povezav na kritični poti, gostota povezav, poraba energije in ostali. Metode *povezovanja* (angl. *routing*) ob upoštevanju načrtovalskih pravil povežejo ustrezne gradnike in pri tem mi-

nimizirajo skupno dolžino povezav. Pri iskanju optimalne razmestitve in povezanosti glede na dane pogoje je večina tovrstnih problemov NP-težkih. Pri snovanju se zato uporabljajo temu primerne tehnike, kot so aproksimacijski algoritmi in različne hevristike. Tehnike iskanja razmestitve obsegajo simulirano ohlajanje [12], analitične metode [13, 14] ter metode na osnovi sil (angl. *force-directed*) [15] in na osnovi razdelitve (angl. *partitioning*) [16]. Za povezovanje se med drugimi uporabljata Leejev algoritem [17] in algoritem A* [18].

Metode za snovanje računalniških vezij pospešujejo razvoj računalnikov, hkrati pa računalnike izkoriščajo za učinkovito delovanje. Zaradi ogromnega števila tranzistorjev na čipu je ročno snovanje velikih sistemov postalo nemogoče. S povečevanjem procesne moči se je razvijalo tudi računalniško podprto snovanje integriranih vezij. Številna programska orodja podpirajo logično in fizično snovanje ter testiranje. Njihov namen je čimbolj avtomatizirati procese modeliranja, simulacije in verifikacije od zasnove vezja do začetka njegove proizvodnje. Tako se lahko zasnujejo in simulirajo vse bolj kompleksna vezja, ki jih je možno fizično implementirati z razvojem proizvodne tehnologije.

Za snovanje vezij z velikim številom gradnikov sta tako kot v primeru CMOS tehnologije tudi pri kvantnih celičnih avtomatih nujni formalizirana metodologija in računalniška podpora [19, 20, 21, 22, 23, 24, 25, 26]. Raziskave na tem področju so smiselne še preden je na voljo tehnologija proizvodnje, saj lahko pripomorejo k njenemu razvoju. Še pred začetkom proizvodnje se določijo računsko uporabni in dejansko izvedljivi sistemi. S simulacijo in analizo alternativnih procesnih platform lahko določimo njihove zmožnosti in jih primerjamo z zmožnostmi obstoječe tehnologije.

Snovanje fizične razmestitve realizacije strukture kvantnih celičnih avtomatov je sestavljeno iz procesov razmeščanja in povezovanja. Pri prvem se na podlagi določenih kriterijev na površino razmeščajo logični primitivi, ki se jih nato v procesu povezovanja poveže z linijami, sestavljenimi iz celic kvantnih celičnih avtomatov. Razmeščanje in povezovanje primitivov kvantnih celičnih avtomatov je v marsičem podobno razmeščanju in povezovanju logičnih gradnikov CMOS vezja. Oba procesa imata enak namen, vendar tudi precejšnje razlike, zato metodologije s področja CMOS ni mogoče neposredno uporabiti na novi platformi. Pomembni razliki sta med drugimi specifičen urin signal [27, 28, 29] in omejeno število križanj linij [30, 31, 32, 33] pri kvantnih celičnih avtomatih. Kljub temu pa je metodologijo zaradi podobnosti možno prilagoditi [34], pri čemer je v pomoč obsežna raziskanost snovanja integriranih vezij. Analiza že razvitih algoritmov

bo pripomogla pri primerjavi in postavljanju standardov za nove metodologije.

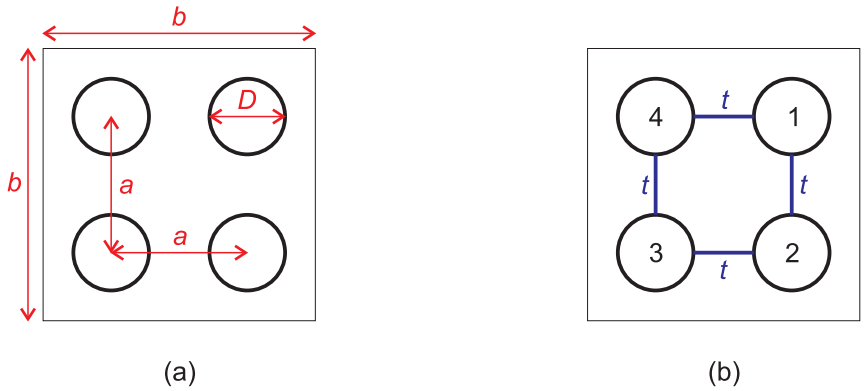
2 Osnove kvantnih celičnih avtomatov

2.1 Standardni model QCA celice

Koncept *QCA celice* (angl. *QCA cell*) sta vpeljala Lent in Tougaw v začetku devetdesetih let prejšnjega stoletja [3]. Standardni model QCA celice sta ista avtorja predstavila v delu [35]. Sestavljajo ga štiri kvantne pike, simetrično razporejene v oglišča kvadrata, kot je shematično prikazano na sliki 2.1(a). Kvantne pike predstavljajo krogi s premerom D. Kvadrat s stranico b, ki vsebuje kvantne pike, označuje logično mejo celice in ni del fizične realizacije [36]. Velikost modelirane QCA celice je določena z velikostjo stranice b. Vodoravna in navpična razdalja med središčema pik je v obeh primerih enaka a. Dolžina stranice kvadrata b je dvakrat večja kot razdalja med središčema kvantih pik [36]. Relacijo med stranico kvadrata in razdaljo med središčema pik izraža enačba

$$b = 2a. \tag{2.1}$$

Če se večje število celic nahaja na določeni površini, potem z izrazom (2.1) določena razdalja b zagotavlja, da med celicami ne bo prišlo do nezaželenih vplivov. Zato jo pri obravnavi QCA celice uporablja več avtorjev [36, 37, 5, 38, 6, 39].



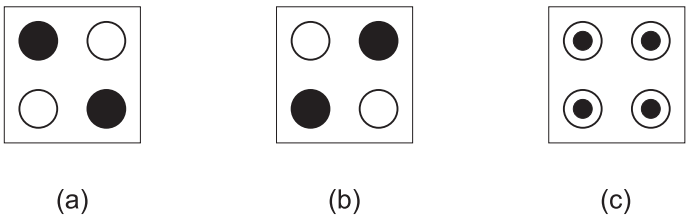
Slika 2.1 Standardni model QCA celice. Slika (a) prikazuje geometrijo celice, slika (b) pa z indeksi označene kvantne pike in tunelirne poti med njimi.

Kvantna pika je področje, v katerem je lahko lokaliziran električni naboj [40]. Obdana je s *pregradami električnega potenciala* (angl. *potential barrier*), ki se lahko spuščajo ali dvigajo. Dovolj visoke potencialne pregrade zadržijo kvantni delec, ki nosi električni naboj, znotraj kvantne pike. V primeru QCA celice je nosilec naboja elektron.

V standardnem modelu QCA celice se nahajata dva mobilna elektrona. Lastnost mobilnosti pomeni, da se lahko ob spuščenih pregradah gibljeta med kvantnimi pikami znotraj celice. Potencialne pregrade obkrožajo tudi vse pike znotraj celice in so dovolj visoke, da elektrona ne moreta uiti izven omenjenih pik [27]. Zato se lahko posamezen elektron nahaja le znotraj ene izmed prostih kvantnih pik v celici, ali pa prehaja med dvema pikama v procesu *kvantnega tuneliranja* (angl. *quantum tunneling*). Kadar sta oba elektrona lokalizirana v kvantnih pikah, je stanje QCA celice določeno z lego elektronov.

Celica je izolirana takrat, ko na elektrone ne delujejo zunanji vplivi. Izolirana celica ima *minimalno energijo*, kadar je razdalja med elektronoma največja, saj je tedaj njuna medsebojna odbojna sila najmanjša. QCA celica z minimalno energijo se nahaja v *osnovnem stanju* (angl. *ground state*) [35]. Elektrona sta medsebojno najbolj oddaljena, ko se nahajata vsak v svoji kvantni piki na eni od dveh diagonal v QCA celici. V standardnem modelu se QCA celica tako lahko nahaja v dveh različnih osnovnih stanjih, ki sta določeni z dvema legama elektronov. Taki legi, pri katerih ima QCA celica minimalno energijo, sta prikazani na slikah 2.2(a) in 2.2(b). V izolirani celici sta obe legi energijsko ekvivalentni in imata zato enako možnost pojavitve. Tedaj je lokalizacija elektrona v katerikoli kvantni piki enako verjetna, kar je shematično prikazano na sliki 2.2(c). QCA

celica se takrat nahaja v nevtralnem stanju [4]. V tem stanju je verjetnost, da je lega elektronov takšna kot na sliki 2.2(a), enaka verjetnosti, da je lega elektronov takšna kot na sliki 2.2(b).

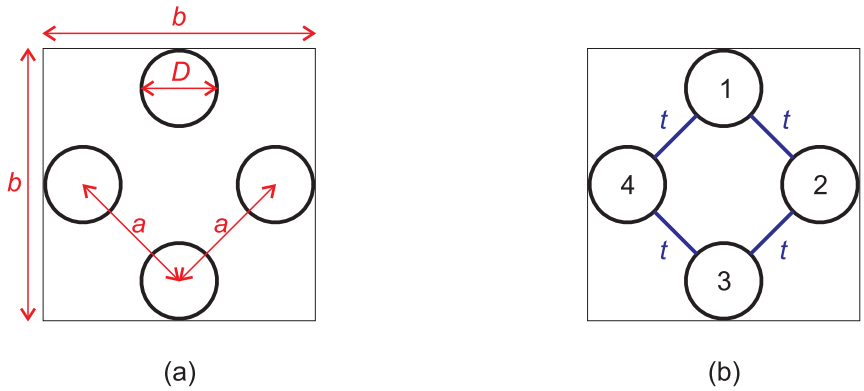


Slika 2.2 Dve energijsko minimalni razporeditvi elektronov v QCA celici ((a), (b)) in nevtralno stanje (c). Gostota naboja v kvantni piki je predstavljena s črnim krogom.

Lent [3] je definiral pojem *polarizacije* (angl. *polarization*), ki označuje stopnjo porazdelitve gostote naboja v konfiguracijah na sliki 2.2. Večja ko je gostota naboja v kvantni piki, bolj verjetno je, da je elektron zajet v njej. Stanju na sliki 2.2(a) je pripisana polarizacija $P = -1$, stanju na sliki 2.2(b) $P = +1$ in stanju na sliki 2.2(c) polarizacija $P = 0$.

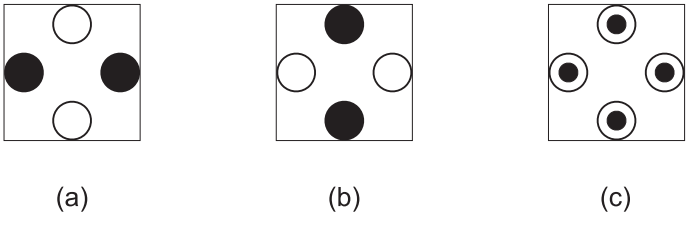
Izolirana QCA celica se nahaja v nevtralnem stanju s polarizacijo $P = 0$ [4]. Na lege in premike elektronov znotraj celice vplivajo notranje in zunanje sile, kot je vpliv Coulombovih sil zaradi prisotnosti elektronov v sosednjih celicah. Takrat se elektrona v celici glede na zunanje vplive razporedita v stanje s polarizacijo -1 ali +1. Stanje celice je možno odčitati in interpretirati kot logično vrednost. Lent [3] je stanju s polarizacijo -1 pripisal logično vrednost 0 in stanju s polarizacijo +1 logično vrednost 1. S tem se osnovni gradnik QCA lahko uporabi za predstavitev dvovrednostno kodiranih podatkov. Vsaka QCA celica tako lahko hrani en bit informacije.

Za procesiranje je poleg predstavitve podatkov potrebno zagotoviti tudi prehajanje med stanji. To je v QCA celici omogočeno s tuneliranjem mobilnih elektronov med kvantnimi pikami. Zaradi zunanjih vplivov lahko elektrona prehajata med sosednjima kvantnima pikama na isti vodoravni premici in med sosednjima pikama na isti navpični premici znotraj QCA celice, ni pa možno tuneliranje po diagonali. Možne tunelirne poti so na sliki 2.1(b) označene s t. Zaradi potencialnih pregrad okrog celice se elektrona stalno nahajata znotraj njenega območja. Z razporeditvijo večjega števila QCA celic na površini in z vpisom vhodnih podatkov v izbrane vhodne QCA celice, je po določenem času možno odčitati stanja izhodnih celic, ki predstavljajo rezultat procesiranja.



Slika 2.3 Rotirana QCA celica. Slika (a) prikazuje geometrijo rotirane celice, slika (b) pa z indeksi označene kvantne pike in tunelirne poti med njimi.

Tougaw [4] je poleg standardnega modela predstavil *rotirano QCA celico* (angl. *rotated QCA cell*), prikazano na sliki 2.3(a). Dve kvantni piki se nahajata na navpični osi in dve na vodoravni osi skozi center celice. Ta celica je identična celici na sliki 2.1, le da je rotirana za 45°. Razdalje z enako oznako na slikah 2.1(a) in 2.3(a) imajo enako dolžino. Če obstaja tunelirna pot med kvantnima pikama z indeksoma i in j na sliki 2.1(b), obstaja tudi pot med pikama z enakima indeksoma na sliki 2.3(b). Za rotirano celico veljajo enake lastnosti in pravila delovanja, kot za standardno QCA celico. Tudi rotirana celica ima najmanjšo energijo, ko je razdalja med elektronoma v njej največja. Eno stanje z minimalno energijo je lega elektronov v kvantnih pikah na navpični osi in drugo energetsko ekvivalentno stanje lega v pikah na vodoravni osi. Obe legi elektronov v rotirani QCA celici in njeno nevtralno stanje so prikazani na sliki 2.4. Tougaw je stanju na sliki 2.4(a) pripisal polarizacijo -1 in logično vrednost 0, stanju na sliki 2.4(b) pa polarizacijo +1 in logično vrednost 1. Na sliki 2.4(c) je prikazano nevtralno stanje s polarizacijo 0.



Slika 2.4 Dve energijsko minimalni razporeditvi elektronov v rotirani QCA celici ((a), (b)) in nevtralno stanje (c).

2.2 Kvantni celični avtomat

QCA je zasnovan na arhitekturi celičnih avtomatov [3]. Celični avtomat je prostorsko in časovno diskreten dinamičen sistem [41], definiran s četverico (L, S, N, f) [42]. Ima določen prostor [41], ki ga predstavlja dvodimenzionalna pravokotna *mreža* (angl. *grid*) L. Končno mrežo velikosti $n \times m$ sestavlja n vrstic in m stolpcev. Element mreže z indeksom i je označen s $celica_i$ in se imenuje *področje* (angl. *site*) ali *celica* (angl. *cell*). Celica $celica_i$ se nahaja v stanju s_i, ki pripada končni množici stanj S. Čas v celičnih avtomatih je diskreten. V časovni točki t se določi novo stanje $s_i(t)$ celice $celica_i$ glede na njeno stanje $s_i(t-1)$ v prejšnji časovni točki $t-1$ in glede na stanja končnega števila sosednjih celic. Sosedstvo celice $celica_r$ z indeksom r je določeno z množico indeksov $N(r)$, za katero velja

$$N(r) = \{i \in L | r - i \in N\}. \tag{2.2}$$

Končna množica N vsebuje indekse, za katere velja

$$\forall c \in N, \forall r \in L : r + c \in L. \tag{2.3}$$

Število sosednjih celic je enako $|N|$. Sosedstvo je vnaprej definirano in obsega celice, ki so običajno nameščene skupaj v prostorskem smislu, zato je celični avtomat *lokalno povezana struktura*. Stanja celic se skozi čas spreminjajo po določenih pravilih, definiranih s preslikavo $f : S^{|N|} \to S$. Stanje celice v času t je določeno s preslikavo $C_t : L \to S$, s katero je za vsako celico v mreži L določeno stanje iz množice S. Sprememba stanja celice z indeksom r je določena z izrazom

$$C_{t+1}(r) = f(\{C_t(i) | i \in N(r)\}). \tag{2.4}$$

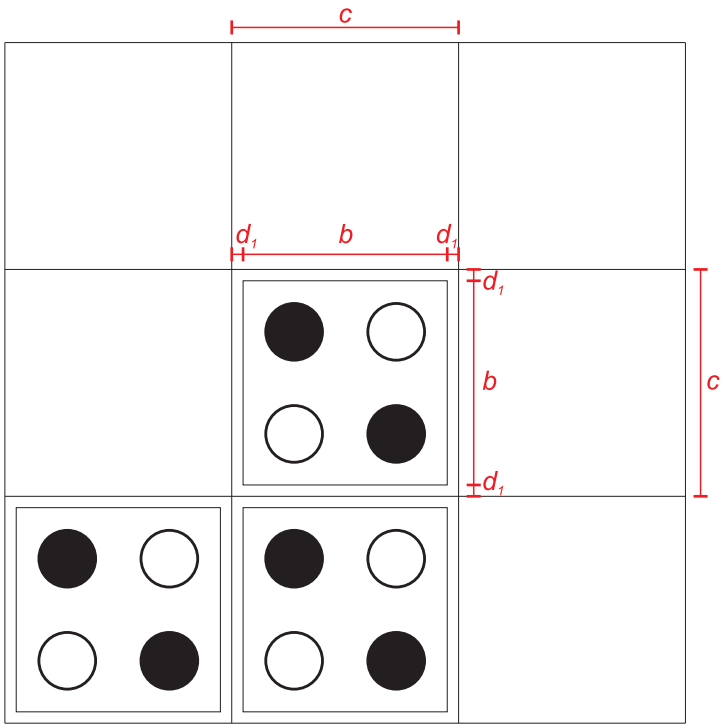
Z modelom celičnega avtomata je možno izvajati procesiranje. Na podlagi vhodnih podatkov se določijo začetna stanja celic v času $t = 0$. Po preteku določenega števila k časovnih korakov se rezultat procesiranja izračuna na podlagi stanj celic v času $t = k$.

Kvantni celični avtomat deluje po podobnem principu kot celični avtomat. Prostor QCA predstavlja pravokotna mreža, sestavljena iz kvadratnih področij. Vsako področje v mreži je lahko prazno, ali pa je na njem nameščena QCA celica. Vsaka QCA celica se lahko nahaja ali v nevtralnem stanju, ali v enem izmed dveh stanj, ki ju interpretiramo kot logični vrednosti 0 in 1. Stanja QCA celic se spreminjajo po zakonih kvantne mehanike. Sosednost je določena z radijem, znotraj katerega ena QCA celica vpliva na ostale.

Zaradi lokalnega vpliva kvantnih fizikalnih zakonov je QCA lokalno povezana struktura.

QCA struktura ima vhodne QCA celice na enem robu pravokotne mreže in izhodne celice na nasprotnem robu. V splošnem naj bodo vhodi na levi strani in izhodi na desni strani mreže. Procesiranje se prične z vpisom začetnega stanja v vhodne QCA celice. Med vhodnimi in izhodnimi celicami se nahajajo notranje celice, ki imajo lahko določeno nespremenljivo *fiksno stanje* z logično vrednostjo 0 ali 1, ali pa so na začetku procesiranja v nevtralnem stanju (slika 2.2(c)). Tok podatkov poteka od vhodnih do izhodnih celic, torej od leve proti desni strani mreže. Po določenem številu korakov se odčitajo stanja izhodnih QCA celic na desnem robu mreže, ki predstavljajo rezultat procesiranja.

Običajno so QCA celice nameščene v središče kvadratnega področja v pravokotni mreži. Slika 2.5 prikazuje tri QCA celice v mreži velikosti 3×3, kjer je vsaka QCA celica nameščena v središče področja v mreži. Stranica QCA celice je označena z b in



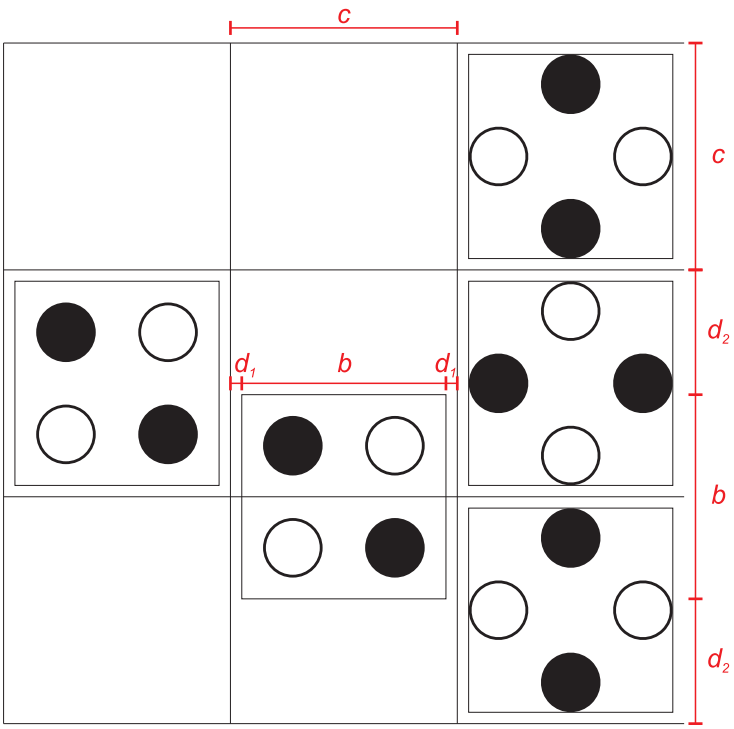
Slika 2.5 Tri QCA celice v mreži velikosti 3×3. Vsaka QCA celica je nameščena v središče področja v mreži. Stranica QCA celice je označena z b, stranica področja v mreži pa s c. Razdalje med istoležnimi robovi QCA celice in področja v mreži so označene z d_1.

stranica področja v mreži s c. Katerokoli področje v mreži lahko vsebuje QCA celico,

zato mora veljati $b \leq c$. Na sliki 2.5 je vsaka QCA celica nameščena v središče področja v mreži, zato so razdalje med njunima levima, desnima, zgornjima in spodnjima roboma enake. Vse naštete razdalje so označene z d_1. Razdalje na sliki 2.5 povezuje enakost

$$c = b + 2d_1. \tag{2.5}$$

Linije v QCA so sestavljene iz zaporedja QCA celic. Sestavljajo jo lahko navadne ali rotirane celice. Na sliki 2.6 sta prikazani vodoravna linija iz navadnih celic v srednji vrstici mreže in navpična linija v desnem stolpcu mreže, ki jo sestavljajo rotirane QCA celice. V tem primeru je za prenos signala iz vodoravne na navpično linijo potrebno zadnjo



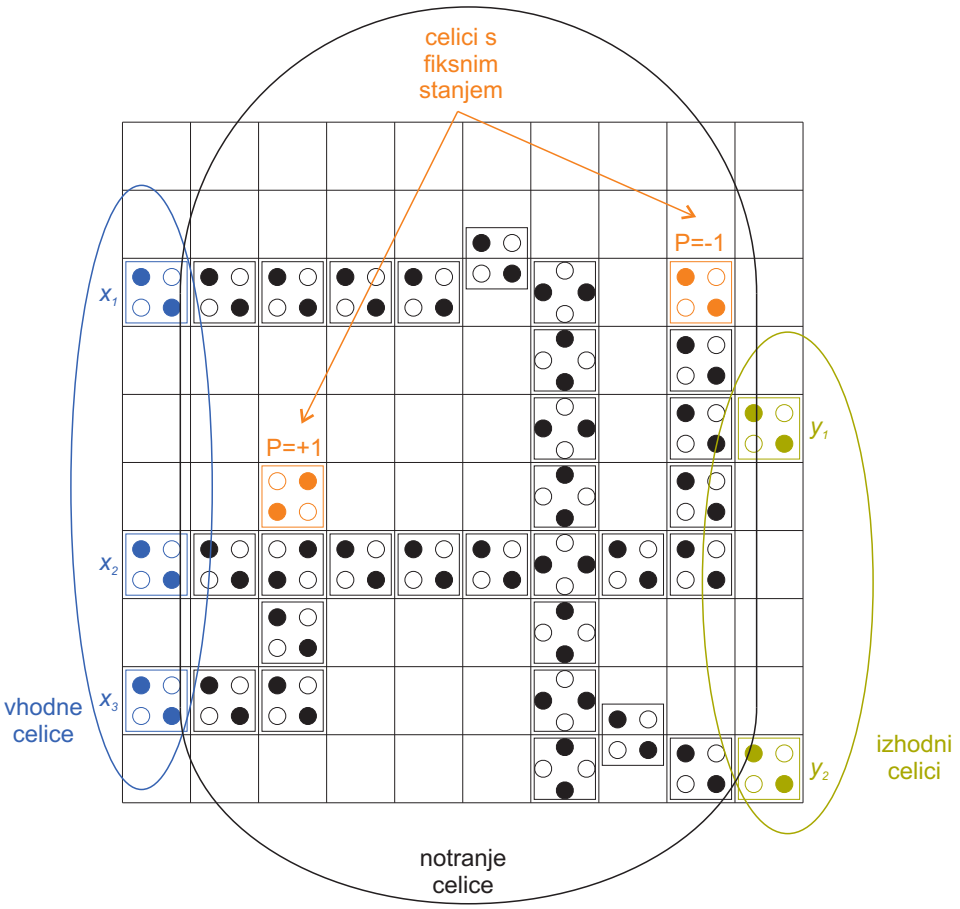
Slika 2.6 Vodoravna linija iz navadnih QCA celic v srednji vrstici mreže in navpična linija iz rotiranih celic v desnem stolpcu mreže. QCA celica v središču mreže je zamaknjena navzdol.

QCA celico v vodoravni liniji zamakniti za polovico stranice področja v mreži navzgor ali navzdol. Tedaj se središče QCA celice nahaja na meji med dvema področjema v mreži. Primer zamaknjene QCA celice je prikazan na sliki 2.6. QCA celica, ki bi se v vodoravni liniji nahajala na področju v središču mreže, je zamaknjena za $\frac{c}{2}$ navzdol. Razdalje z enako oznako na slikah 2.5 in 2.6 imajo enako dolžino. Na sliki 2.6 je razdalja

med levima robovoma področja v mreži in QCA celice na njem enaka d_1, tako kot na sliki 2.5. Podobno je tudi razdalja med desnima robovoma enaka d_1. Razdalja med zgornjim robom zamaknjene QCA celice in zgornjim robom področja v središču mreže je označena z d_2. Enako dolžino ima razdalja med spodnjim robom zamaknjene QCA celice in spodnjim robom področja na dnu srednjega stolpca v mreži. Za količine na sliki 2.6 poleg enačbe (2.5) velja tudi

$$b + 2d_2 = 2c. \tag{2.6}$$

Na sliki 2.7 je prikazan preprost QCA, sestavljen iz petintridesetih QCA celic, ki so nameščene na področja v mreži velikosti 10×10. QCA vsebuje tri vhodne celice,



Slika 2.7 Primer preprostega QCA.

označene z x_1, x_2 in x_3. Dve izhodni celici sta označeni z y_1 in y_2, ostale QCA celice

pa so notranje. V vhodne celice se vpišejo vhodni podatki in po določenem času se iz izhodnih celic odčita rezultat procesiranja. Notranje celice so razdeljene na dve vrsti [4]:

- celice s fiksno določenim začetnim stanjem: takšne celice imajo stalno polarizacijo -1 ali +1 in so v QCA nameščene z namenom, da neprestano vplivajo na sosednje celice;
- delovne celice, katerih stanje se med procesiranjem lahko spreminja: med procesiranjem se njihova stanja določijo glede na vplive sosednjih QCA celic, tako da je skupna energija QCA minimalna.

Procesiranje poteka od vhodnih QCA celic na levem robu mreže do izhodnih celic na njenem desnem robu. Razporeditev elektronov v notranjih celicah sčasoma določi stanje v izhodnih celicah.

2.3 Nadzor preklopa stanja v QCA celici z urinim signalom

Kvantnomehanski sistem se nahaja v osnovnem stanju takrat, ko ima najmanjšo možno energijo. Vsako stanje z večjo energijo je *vzbujeno stanje* (angl. *excited state*). Slednje je običajno kratkotrajno, ker sistem z oddajanjem energije prehaja proti osnovnemu stanju. Lahko pa sistem doseže lokalni energetski minimum, ki ni enak minimalni možni energiji. Tedaj se nahaja v dolgotrajnem vzbujenem stanju, imenovanem *metastabilno stanje* (angl. *metastable state*).

Ob predstavitvi QCA je Lent [3] uvedel koncept *procesiranja z osnovnim stanjem* (angl. *computing with the ground state*). Z vpisom stanja v vhodne QCA celice se v sistem vnese energija in ta preide iz začetnega osnovnega v vzbujeno stanje z večjo energijo. Po času t se QCA ustali v končnem osnovnem stanju, iz katerega se razbere rezultat procesiranja. Princip procesiranja z osnovnim stanjem določa, da je za rezultat procesiranja pomembno le končno osnovno stanje sistema, ki je neodvisno od mehanizmov oddajanja energije ob prehodu iz vzbujenega v osnovno stanje. Sprva je Lent predlagal, da se začetna stanja nenadno vpišejo v vhodne QCA celice in se jih nato ne spreminja, dokler se sistem ne ustali v končnem osnovnem stanju. Takšno prehajanje stanj imenujemo za *grobo preklapljanje* (angl. *abrupt switching*) z *disipacijo energije v okolje* (angl. *dissipative coupling to the environment*).

Pri preučevanju dinamike delovanja QCA z grobim preklapljanjem sta Tougaw in Lent [35] naletela na problem metastabilnih stanj. Med grobim preklapljanjem stanj lahko

sistem zaide v dolgotrajno metastabilno stanje, kar nepredvidljivo poveča čas umirjanja t. Vendar uporabnost QCA s stališča procesiranja zahteva, da se sistem po vnaprej znanem konstantnem času ustali v osnovnem stanju, ki edino določa logično pravilen rezultat procesiranja.

Rešitev z uporabo *adiabatnega preklapljanja* (angl. *adiabatic switching*) sta avtorja predlagala v [27]. Pri tem pristopu je prehajanje stanj nadzorovano in poteka postopno. Začetna stanja se v vhodne celice ne vpišejo nenadno, ampak vpis poteka zadosti počasi, kar prepreči pojav metastabilnih stanj. Postopno prehajanje sestavlja več faz, v katerih se iz QCA celice odstrani staro stanje in se jo pripravi za določitev novega. Takšen preklop stanja se lahko izvede v katerikoli časovni točki in se lahko med procesiranjem večkrat ponovi. Če je preklop izveden dovolj počasi, sistem gladko preide iz predhodnega stanja v trenutno osnovno stanje. Adiabatno preklapljanje tako zagotavlja, da se QCA stalno nahaja v trenutnem osnovnem stanju, določenim s stanjem vhodnih QCA celic in trenutno fazo preklopa. S tem je preprečena možnost, da bi sistem QCA zašel v metastabilno stanje [27].

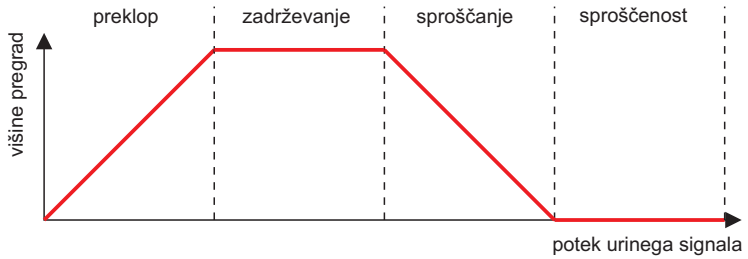
Kontrola preklopa se izvaja z nadzorom tuneliranja mobilnih elektronov med kvantnimi pikami v QCA celici, implementiranim z *urinim signalom* (angl. *clock signal*), ki določa višine potencialnih pregrad med pikami. Pri dvignjenih pregradah je tuneliranje onemogočeno, pri spuščenih pregradah pa lahko elektrona prosto tunelirata med pikami. V fazi onemogočenega tuneliranja se stanje QCA celice ne more spremeniti. V tej fazi celica služi kot vhodna QCA celica, ki s svojim zaklenjenim stanjem vpliva na spremembo stanja sosednjih celic. Ko je potrebno spremeniti stanje celice glede na njene sosede, mora urin signal preiti v fazo omogočenega tuneliranja.

Urin signal vsebuje fazo z dvignjenimi potencialnimi pregradami in fazo s spuščenimi pregradami. Prehod med tema dvema fazama ni izveden v trenutku, pač pa med njima potekata še fazi spuščanja in dviganja pregrad. Tako torej urin signal skupno sestavljajo štiri *urine faze* (angl. *clock phase*). Višine potencialnih pregrad med kvantnimi pikami v posamezni urini fazi so ponazorjene z diagramom na sliki 2.8. Štiri faze QCA ure, kot jih je predstavil in poimenoval Lent [27], so:

- **Faza preklopa** (angl. *switch phase* - *S*): Na začetku faze preklopa so potencialne pregrade spuščene, kar omogoča tuneliranje elektronov med kvantnimi pikami v QCA celici. Takrat se celica nahaja v nevtralnem stanju. Skozi fazo se pregrade

dvigujejo, dokler ne dosežejo zgornje meje, pri kateri je tuneliranje onemogočeno. Polarizacija celice prehaja od $P = 0$ proti določeni vrednosti $P = -1$ oziroma $P = +1$, ki je odvisna od polarizacije vhodnih in sosednjih celic. V tej fazi se torej vrši dejansko procesiranje.

- **Faza zadrževanja** (angl. *hold phase* - H): Višina potencialnih pregrad je stalno na najvišji ravni, zato se razporeditev elektronov v QCA celici ne spreminja. Celica lahko služi kot vhodna celica za sosede, v katerih preklop stanja kontrolira urin signal v fazi preklopa. Po drugi strani pa ima lahko celica vlogo izhodne celice celotnega QCA, pri čemer je možno razbrati njeno stanje in ga interpretirati kot logično vrednost.
- **Faza sproščanja** (angl. *release phase* - R): V tej fazi se potencialne pregrade spuščajo in celica ponovno preide v nevtralno stanje. Faza sproščanja služi izbrisu starega stanja iz QCA celice.
- **Faza sproščenosti** (angl. *relaxed phase* - L): Potencialne pregrade so spuščene in elektrona lahko neovirano tunelirata med kvantnimi pikami. QCA celica ostaja v nevtralnem stanju s polarizacijo $P = 0$.



Slika 2.8 Višine potencialnih pregrad med kvantnimi pikami znotraj QCA celice v posamezni fazi QCA urinega signala.

Vse urine faze trajajo enako dolgo in si sledijo v zaporedju, prikazanem na sliki 2.8. Urin signal je cikličen, tako da si vse štiri urine faze periodično sledijo v navedenem zaporedju. Minimalni čas trajanja enega cikla T_{cikel} je sorazmeren s številom celic N, ki jih kontrolira urin signal [27]:

$$T_{cikel} \propto N. \tag{2.7}$$

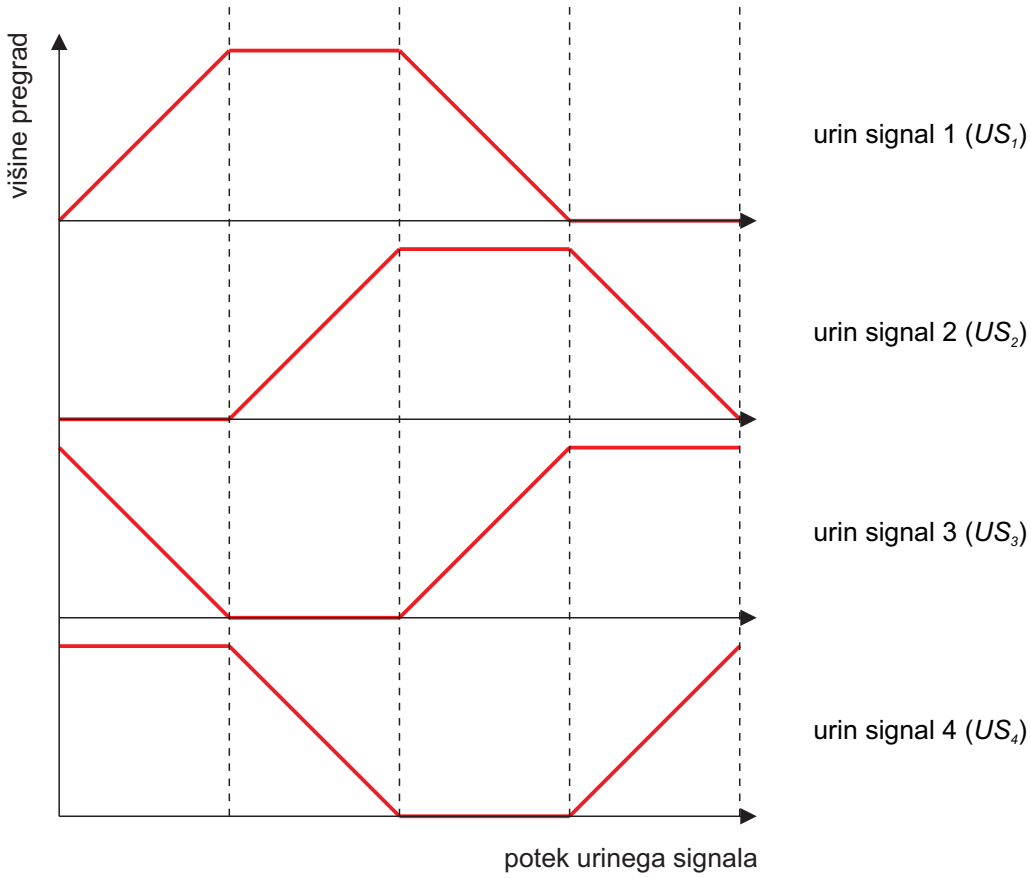
Tem več QCA celic kontrolira urin signal, daljša mora biti njegova perioda, tako da bo zaporedno procesiranje potekalo pravilno.

Generiranje urinega signala omogoča elektroda, ki je nameščena pod plastjo substrata s QCA celicami. Ena elektroda lahko kontrolira eno samo QCA celico ali pa večjo skupino celic. Poleg tega se lahko pod plastjo substrata nahaja večje število elektrod, od katerih vsaka kontrolira svojo skupino celic. *Urina cona* (angl. *clocking zone*) je omejeno območje v mreži, sestavljeno iz množice sosednjih kvadratnih področij. Določena je z območjem, pod katerim se nahaja ena elektroda. Vsaka elektroda določa natanko eno urino cono. Preklope stanj vseh QCA celic v isti urini coni nadzoruje ista elektroda.

Na podlagi navedenih dejstev je Lent zasnoval *cevovodno* (angl. *pipelined*) arhitekturo QCA z adiabatnim preklapljanjem, v kateri je QCA razdeljen na particije. Med particijami in urinimi conami obstaja bijektivna preslikava, tako da vse QCA celice v isti particiji spadajo v isto urino cono. Posamezna elektroda sedaj generira enega od štirih urinih signalov, ki so med seboj fazno zamaknjeni. Ko se prvi signal nahaja v fazi preklopa, je drugi v fazi sproščenosti, tretji v fazi sproščanja in četrti v fazi zadrževanja. Vsak signal posebej še vedno poteka v zaporedju, ki je prikazano na sliki 2.8. Sočasen potek štirih medsebojno fazno zamaknjenih urinih signalov ponazarja graf na sliki 2.9.

Vsaka particija v cevovodni arhitekturi QCA deluje kot procesni sistem z vhodi in izhodi. Vzemimo za primer sosednje particije P_1, P_2 in P_3. Preklope stanj QCA celic v P_1 nadzoruje urin signal US_1, preklope stanj v P_2 signal US_2 in preklope v P_3 signal US_3 . Ko se US_1 nahaja v fazi zadrževanja in US_2 v fazi preklopa, služijo celice v P_1 blizu skupnega roba s P_2 kot vhodne celice za P_2. Stanja celic v P_2 se določijo glede na stanja vhodnih celic. Signal US_3 je v fazi sproščenosti, zato da ne moti prehajanja stanj celic v P_2. Tako se lahko s celicami v P_2 izoblikuje vmesni rezultat procesiranja. V naslednjem časovnem koraku je US_1 v fazi sproščanja, US_2 v fazi zadrževanja in US_3 v fazi preklopa. Sedaj so celice v P_2 blizu skupnega roba s P_3 vhodne celice za P_3. Opisani postopek se ponavlja, dokler končni rezultat procesiranja ni zapisan v izhodnih celicah celotnega QCA.

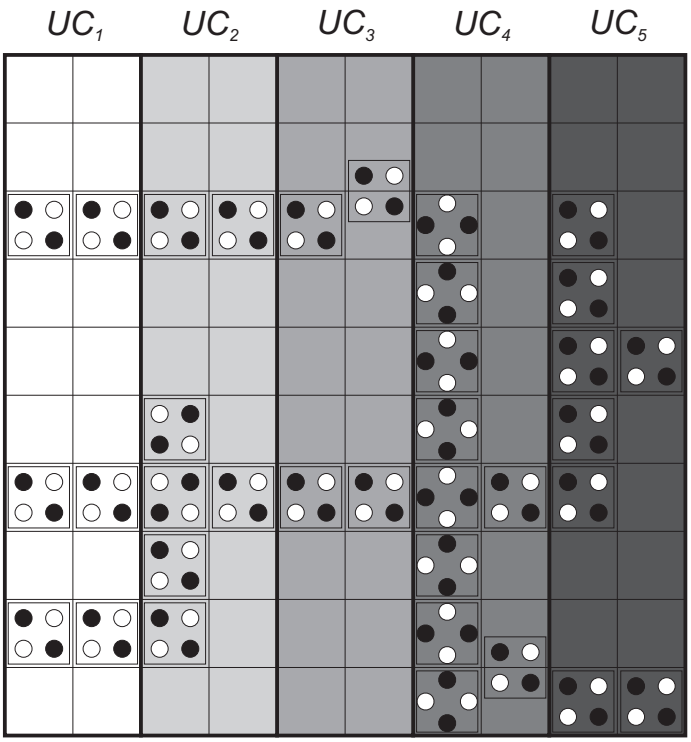
V primeru na sliki 2.10 je mreža razdeljena na pet urinih con UC_1, UC_2, UC_3, UC_4 in UC_5. Tudi QCA je razdeljen na pet particij P_1, P_2, P_3, P_4 in P_5, tako da je vsaka QCA celica v particiji P_i nameščena na neko področje v urini coni UC_i. Za pravilno smer toka podatkov od vhodnih do izhodnih celic QCA morajo elektrode pod zaporednimi urinimi conami generirati ustrezno zamaknjene urine signale. Vzemimo, da prehajanja stanj v UC_1 nadzoruje signal US_1, v UC_2 signal US_2, v UC_3 signal US_3 in v UC_4 signal US_4. Zaradi cikličnosti mora preklop v coni UC_5 nadzorovati signal US_1. Naj bo signal US_1



Slika 2.9 Potek štirih medsebojno fazno zamaknjenih urinih signalov. Signal 2 je za eno fazo zamaknjen od signala 1, signal 3 je za eno fazo zamaknjen od signala 2 in za dve fazi od signala 1, itd.

v fazi preklopa. Tedaj mora biti US_2 v fazi sproščenosti, US_3 v fazi sproščanja in US_4 v fazi zadrževanja.

Cevovodno delovanje omogoča sproščanje QCA celic, ki so že opravile procesiranje in vmesni rezultat posredovale celicam v naslednji particiji. Proste celice se uporabijo za nadaljnje procesiranje. Kadar se urin signal v k particijah nahaja v fazi preklopa, se v QCA sočasno izvaja k neodvisnih procesiranj. S tem se poveča prepustnost celotnega QCA. Zakoni termodinamike omejujejo število QCA celic, ki lahko hkrati preidejo v novo logično pravilno stanje. Zato je potrebno QCA celice porazdeliti v več manjših skupin, v katerih lahko vse celice hkrati preidejo v novo stanje. Razdelitev QCA v urine cone dopušča sočasni preklop manjšim skupinam celic in tako povečuje možnost uporabe QCA. Z uporabo QCA ure se lahko določi smer toka podatkov v QCA. Ura služi tudi za



Slika 2.10 Primer cevovodne arhitekture QCA. Mreža velikosti $10c \times 10c$ je razdeljena na pet urinih con, poimenovanih UC_1, UC_2, UC_3, UC_4 in UC_5. Vse imajo enako obliko pokončnega pravokotnika s širino $2c$ in višino $10c$. Področja v isti coni so obarvana z isto barvo. Tok podatkov poteka od levega proti desnemu robu mreže, torej od UC_1 do UC_5.

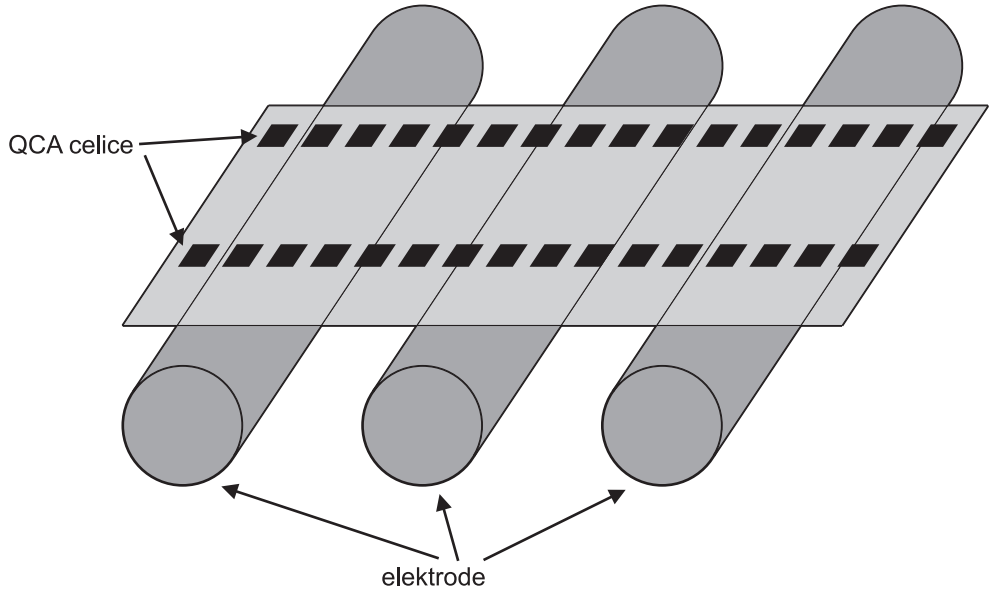
ojačitev signala, ki poteka po QCA celicah [43, 44]. Energija signala se manjša zaradi disipacije v okolje. Če signal nima dovolj energije, pride do izgube informacije. Ura dovaja energijo tako, da pri preklopu obnovi stanje v popolno polarizacijo. S tem se tudi popolnoma obnovi informacija, ki jo signal nosi po QCA celicah.

2.4 Implementacija QCA ure

Vpeljavo QCA urinega signala sta predlagala Lent in Tougaw v [27]. Generirajo ga pod QCA celicami nameščene elektrode, ki ustvarjajo električno polje. Smer in jakost polja določata višino potencialnih pregrad med kvantnimi pikami znotraj celice [29]. Ciklično spreminjanje smeri in jakosti polja ustreza cikličnemu QCA signalu s štirimi fazami.

Urin signal generirajo elektrode, ki z ustvarjanjem električnega oziroma magnetnega polja določajo višino potencialnih pregrad med kvantnimi pikami. Elektrode se nahajajo

pod plastjo substrata s QCA celicami [29, 45], kot je prikazano na sliki 2.11.



Slika 2.11 Elektrode, ki ustvarjajo električno oziroma magnetno polje, potekajo pod plastjo substrata s QCA celicami. Vse QCA celice nad isto elektrodo pripadajo isti urini coni.

Elektrode pod plastjo substrata s QCA celicami sestavljajo vezje, ki omogoča ustvarjanje ustrezno fazno zamaknjenega urinega signala v vsaki urini coni. Tako vezje je mogoče zasnovati z uporabo obstoječih načrtovalskih pravil za snovanje CMOS vezja. Vezje je lahko izdelano iz silicijevih elektrod, položenih na plast, ki je nameščena pod plastjo substrata in poteka vzporedno z njo [20]. Kompleksno urino vezje pa lahko onemogoči prednosti, ki jih ponuja QCA tehnologija glede na sedanje CMOS sisteme. Počasno kompleksno urino vezje ne bi dovoljevalo izkoriščanja visoke hitrosti preklopa stanj v QCA celicah. Poleg tega je lahko širina elektrode nekajkrat večja od velikosti uporabne QCA celice. Zaradi teh dejstev je potrebno pri snovanju QCA upoštevati tudi realizacijo vezja za generiranje urinega signala. Kot alternativno implementacijo urinega vezja so predlagali realizacijo z ogljikovimi nanocevkami s širino velikostnega reda nanometra [46], za kar pa mora tehnologija še dozoreti.

Realizacijo z uro nadzorovane QCA celice na osnovi kovinskih otokov je predlagal Toth [47] in eksperimentalno izvedel Orlov s sodelavci [48]. V ta namen so izdelali celico s šestimi kovinskimi otoki. Shemo urinega vezja za nadzor preklopa molekularnih QCA celic je predstavil Hennessy [29]. V objavljeni shemi urin signal omogoča električno polje,

ki ga ustvarjajo žice pod QCA celicami. Vpeljavo ure v molekularni QCA obravnava tudi Lent [49]. Pri magnetni implementaciji QCA je urin signal realiziran z uporabo magnetnega polja [50]. V [51] so avtorji eksperimentalno izdelali z uro nadzorovana logična vrata v tehnologiji magnetnega QCA. Eksperimente z uporabo bakrenih elektrod s širinami od 1 do 2,4 μm in dolžino 500 μm so predstavili v delu v [45]. Pri tem so uporabili QCA celice velikosti $57 \times 95 \times 30$ nm in $62 \times 102 \times 30$ nm.

2.5 Fizična realizacija QCA celice

Vsak logični primitiv, ki je zgrajen na osnovi QCA, je sestavljen izključno iz QCA celic. Za praktično uporabo je seveda nujno QCA celico fizično izdelati. Ob predstavitvi modela koncepta QCA je Lent [3] predlagal realizacijo celice s polprevodniki, kasneje pa so se pojavile nove ideje za izdelavo. Realizacija QCA celice je odvisna od tehnologije implementacije QCA. Vse realizacije ne temeljijo na standardnem modelu QCA celice, vendar delujejo po istem principu. Vsem je skupno dejstvo, da se celica lahko nahaja v različnih stanjih. Posamezna QCA celica se vedno nahaja v natanko enem stanju. Katerokoli stanje je možno vpisati v vhodno celico in ga odčitati iz izhodne celice. Stanje QCA celice se lahko spremeni zaradi notranjih in zunanjih vplivov, pravila prehoda iz starega v novo stanje pa določajo zakoni kvantne mehanike.

Danes so splošno znani štiri različni tipi realizacije QCA celice. Vsaka od njih ima svoje prednosti in slabosti, ki so orisane v naslednjih alinejah:

- **Polprevodniški QCA** (angl. *semiconductor QCA*): Uporaba polprevodnikov [3] je bila prva ideja za realizacijo QCA celice. Njena prednost je v možnosti izkoriščanja visoko razvite tehnologije izdelave polprevodniških naprav [52]. Lent je predlagal realizacijo standardne QCA celice velikosti velikostnega reda 10 nm v *GaAs/AlGaAs* sistemu, v katerem so tudi izvedli eksperimente [53, 54]. Realizirali so tudi QCA celico na osnovi silicija [55]. Do tega trenutka tehnologija še ne omogoča masovne proizvodnje polprevodniške QCA celice takšnega velikostnega reda. Številni avtorji uporabljajo polprevodniško QCA celico kot prototipni model v svojih raziskavah. Lent [3] je prve izračune izvedel na modelu s parametri $a = 20$ nm, $b = 40$ nm in $D = 10$ nm.
- **Realizacija s kovinskimi otoki** (angl. *metal-island QCA*): V tej realizaciji je kvantna pika izdelana iz aluminijevega otoka. Dve piki, med katerima sme tuneli-

rati elektron, sta povezani z $Al/AlO_x/Al$ vezjo. Izdelavo QCA celice s kovinskimi otoki je predlagal Lent [56], realiziral pa jo je Orlov s sodelavci [57]. S predstavljeno realizacijo so izvedli več eksperimentov [58, 59, 60, 61, 62, 63]. Opisana realizacija je bila izvedena predvsem kot dokaz, da je možno QCA celice fizično izdelati. Realizacija ima namreč zelo omejene možnosti skalabilnosti. Aluminijev otok je v velikostnem redu enega μm, zato lahko celica deluje le pri izjemno nizki temperaturi blizu absolutne ničle. Amlani [58, 64] je izvedel eksperiment s parametri $a = 3$ μm, $b = 6$ μm in $D = 1,4$ μm.

- **Molekularni QCA** (angl. *molecular QCA*): Molekularno realizacijo sta predlagala Lent in Tougaw [35]. Vsaka celica je zgrajena iz ene same molekule velikostnega reda 1 nm [64, 65, 66]. Tako majhna celica prinaša pomembne prednosti molekularne realizacije pred ostalimi. Med prednostmi so velika gostota QCA celic v sistemu, velika hitrost preklopa stanja in delovanje pri sobni temperaturi. Molekularna realizacija prinaša tudi naravno simetričnost celice in možnost masovne izdelave s postopkom *samosestavljanja* (angl. *self-assembly*). Trenutno še ni razvite tehnologije, ki bi omogočala nameščanje posamezne molekule in interakcijo med molekulo in okoljem. Lent [66] je predlagal uporabo molekule 1,4-dialil butan radikal kation, pri kateri so določene vrednosti $a = 0,7$ nm, $b = 1,4$ nm in $D = 0,5$ nm.

- **Magnetni QCA** (angl. *magnetic QCA*): Idejo realizacije z nanomagneti sta vpeljala in jo eksperimentalno realizirala Cowburn in Welland [67]. QCA celica je v tem primeru sestavljena iz enega samega nanomagneta velikosti od 10 do 100 nm [50, 51, 68]. Magnetna celica ne sovpada več s standardnim modelom, še vedno pa deluje po enakem principu. Celica se lahko nahaja v dveh različnih stanjih, določenih z magnetizacijo nanomagneta. Elektrostatične sile, ki v standardnem modelu omogočajo interakcijo med celicami, so v magnetnem modelu nadomeščene z magnetnimi silami. Prednost magnetne realizacije je v tem, da lahko tudi relativno velike QCA celice delujejo pri sobni temperaturi, njena slabost pa je počasen preklop. Ker je magnetna QCA celica v celoti enaka nanomagnetu, parametri a, b in D tu nimajo pomena. V delu [68] so avtorji modeliranje osnovali na QCA celici, določeni z nanomagnetom v obliki kvadra velikosti 20 nm $\times$ 60 nm $\times$ 120 nm.

Fizično je QCA sestavljen iz več plasti. Na spodnji plasti se nahaja vezje, ki generira

urin signal. Nad njim je plast substrata, na katerem so nameščene QCA celice. V večnivojski realizaciji QCA [69] se nad prvo plastjo substrata nahaja še več enakih plasti. Večnivojski QCA je sestavljen iz štirih plasti substrata. Na prvi plasti QCA celice sestavljajo logično vezje. Naslednji dve plasti sta namenjeni za *vertikalno povezavo* (angl. *via*) od spodnje do najbolj zgornje plasti substrata. Vertikalno povezavo sestavljajo navadne QCA celice. Zaradi predstavitve v dvodimenzionalnem prostoru jih označujemo s simboli s slike 2.12(a). Vse celice v vertikalni povezavi se nahajajo na navpični osi, ki poteka

Slika 2.12 Prikaz celice v vertikalni povezavi (a). Prikaz celice na zgornji plasti (b).

od spodnje do zgornje plasti substrata. Najbolj zgornja plast substrata je namenjena za nameščanje QCA celic v linijah, ki se križajo z linijami na prvi plasti. Celice na zgornji plasti označujemo s prikazom na sliki 2.12(b).

2.6 Osnovne fizikalne lastnosti QCA

Velikost električnega naboja enega elektrona, označena s $q_{elektron}$, je podana z enačbo

$$q_{elektron} = -e, \tag{2.8}$$

kjer je e elementarni naboj. Vrednost elementarnega naboja je približno 1,602 $\times$ 10^{-19} C. Ker mora biti QCA celica elektrostatično nevtralna [70], je vsaki kvantni piki prirejen naboj q_{pika} z velikostjo

$$q_{pika} = \frac{e}{2}. \tag{2.9}$$

Vsota nabojev v celici s štirimi kvantnimi pikami in dvema elektronoma je enaka

$$4q_{pika} + 2q_{elektron} = 0, \tag{2.10}$$

kar zagotavlja elektrostatično nevtralnost celice.

Vsaka kvantna pika je označena z indeksom iz množice $\{1, 2, 3, 4\}$. Razmestitev z indeksi označenih pik v standardnem modelu QCA celice je prikazana na sliki 2.1(b),

razmestitev pik v rotirani celici pa na sliki 2.3(b). Skupen naboj q_i pike i je odvisen od prisotnosti elektrona v tej piki. Kadar je elektron popolnoma lokaliziran v piki i, je njen naboj

$$q_i = q_{pika} + q_{elektron} = -\frac{e}{2}, \tag{2.11}$$

če pa ni mogoče, da se elektron nahaja v tej piki, je njen naboj enak q_{pika}:

$$q_i = q_{pika} = \frac{e}{2}. \tag{2.12}$$

Gostota naboja ρ_i v kvantni piki i se izračuna z enačbo

$$\rho_i = \frac{q_i - q_{pika}}{-e}. \tag{2.13}$$

Vrednosti gostote se nahajajo na intervalu [0, 1]. V robnih primerih, ko je verjetnost lokalizacije elektrona v piki i enaka 0 oziroma 1, je tudi gostota naboja ρ_i enaka 0 oziroma 1. Polarizacija P v standardnem modelu QCA celice je po [70] definirana kot

$$P = \frac{(\rho_1 + \rho_3) - (\rho_2 + \rho_4)}{\rho_1 + \rho_2 + \rho_3 + \rho_4}. \tag{2.14}$$

Procesiranje z osnovnim stanjem je zelo občutljivo na temperaturo okolja. Če termične fluktuacije povzročijo povečanje skupne energije celic v QCA, tako da je ta večja od energije njegovega osnovnega stanja, se lahko stanja celic neželeno spremenijo, kar privede do logično nepravilnega rezultata. *Energija vzbujanja* (angl. *excitation energy*) je zunanja energija, ki povzroči neželeno spremembo stanja celice. Pri robustnem delovanju QCA ne sme priti do neželenih zunanjih vplivov že pri majhni energiji, zato mora biti energija vzbujanja čim večja. Želeno je, da je njena vrednost veliko večja od k_bT [27], kjer je $k_b = 1,381 \times 10^{-23}$ J/K Boltzmannova konstanta in T temperatura okolja v kelvinih. Manjša kot je QCA celica, večja je razlika med energijama osnovnega in vzbujenega stanja. Zato je večja energija vzbujanja, s tem pa je možna višja temperatura okolja, v katerem deluje QCA. Lent je izračunal maksimalno temperaturo, pri kateri naj bi standardna polprevodniška QCA celica z razdaljo med kvantnima pikama $a = 20$ nm še delovala. Ta znaša komaj 7 K, medtem ko bi lahko molekularna celica z $a = 2$ nm delovala tudi pri 700 K.

Lent [27] je predstavil primer linije, sestavljene iz N zaporedno nameščenih QCA celic, po kateri se prenaša signal z logično vrednostjo 1. V osnovnem stanju linije se vse celice nahajajo v stanju z logično vrednostjo 1. Pri konfiguraciji v vzbujenem stanju se prvih m celic v liniji nahaja v stanju z vrednostjo 1 in ostalih $N-m$ celic v stanju z vrednostjo 0.

Energijo vzbujanja, ki je potrebna za neželeno spremembo stanja celice in s tem prehod iz osnovnega v vzbujeno stanje, je Lent označil z E_k. Slednja je neodvisna od vrednosti m in dolžine linije N, se pa z večjim N veča število možnih napačnih konfiguracij in s tem entropija vzbujenega stanja. Lent je izračunal zgornjo mejo števila QCA celic v liniji znotraj ene urine cone N_{max}, ki znaša

$$N_{max} = e^{E_k/k_b T}. \tag{2.15}$$

2.7 Modeliranje QCA

Za modeliranje in simulacijo delovanja QCA je bilo razvitih več različnih modelov. Med seboj se razlikujejo v bolj ali manj natančni fizikalni obravnavi sistema QCA. Nekateri poenostavljeno upoštevajo le zakone klasične mehanike, drugi pa vključujejo tudi kompleksnejšo kvantnomehansko analizo. Nekateri modeli ne omogočajo simulacije dinamičnega delovanja sistema, temveč le verifikacijo logične funkcionalnosti. Simulacije obnašanja različnih modelov imajo različne računske kompleksnosti. Za analizo modelov z eksponentno zahtevnostjo se uporabljajo tehnike z manjšo računsko kompleksnostjo, s katerimi se izračunajo približne vrednosti dejanskega rezultata. V nadaljevanju so opisani trije modeli QCA in sicer polklasični model, bistabilna aproksimacija in model na osnovi koherenčnega vektorja.

Eden izmed enostavnih modelov je *polklasični model* (angl. *semiclassical model*), ki ga je predstavil Macucci [71]. Elektrone obravnava kot klasične delce, vendar pri tem omogoča njihovo tuneliranje med kvantnimi pikami, kar je posebnost kvantne mehanike. Elektrostatična energija QCA je v polklasičnem modelu izražena z energijo sistema točkovnih nabojev

$$E = \sum_{i \neq j} \frac{q_i q_j}{4\pi\varepsilon_0 \varepsilon_r r_{ij}}, \tag{2.16}$$

kjer je q_i skupen naboj pike i, q_j skupen naboj pike j, $\varepsilon_0 = 8,854 \times 10^{-12}$ F/m dielektričnost vakuuma, ε_r relativna dielektričnost medija in r_{ij} razdalja med pikama i in j. S pregledom vseh možnih kombinacij vrednosti nabojev v pikah se izračuna minimalna energija. Tako se ugotovi, kdaj se sistem nahaja v osnovnem stanju in kakšna je takrat kombinacija vrednosti nabojev v pikah. Na podlagi slednje se določi polarizacije celic v simuliranem QCA. Zaradi eksponentne časovne zahtevnosti je simulacija z uporabo polklasičnega modela primerna le za QCA z majhnim številom celic, za simuliranje velikih

QCA pa se uporabljajo hevristične metode.

Bistabilni modeli (angl. *bistable models*) poenostavljeno obravnavajo vsako QCA celico kot dvostanjski sistem. V takih modelih imata celici i in j ali enako ali pa nasprotno polarizacijo. Elektrostatična energija med kvantno piko v celici i in piko v celici j je podana z enačbo

$$E_{i,j} = \frac{q_i q_j}{4\pi\varepsilon_0\varepsilon_r r_{ij}}, \tag{2.17}$$

kjer je q_i skupen naboj pike v celici i, q_j skupen naboj pike v celici j, ε_0 dielektričnost vakuuma, ε_r relativna dielektričnost medija in r_{ij} razdalja med pikama i in j. Vsota izraza (2.17) po vseh pikah v celicah i in j določa energijo sistema celic i in j:

$$\hat{E}_{i,j} = \sum_{i,j} E_{i,j}. \tag{2.18}$$

Z uporabo izrazov (2.17) in (2.18) se izračunata energija $\hat{E}^1_{i,j}$ sistema celic i in j z isto polarizacijo in energija $\hat{E}^2_{i,j}$ sistema celic i in j z nasprotnima polarizacijama. Energija $E^k_{i,j}$ se izračuna kot razlika

$$E^k_{i,j} = \hat{E}^2_{i,j} - \hat{E}^1_{i,j}. \tag{2.19}$$

Za vsako dvostanjsko celico i se lahko določi Hamiltonova matrika H_i, ki predstavlja njeno energijo [72]:

$$H_i = \sum_j \begin{bmatrix} -\frac{1}{2}P_j E^k_{i,j} & -\gamma_i \\ -\gamma_i & \frac{1}{2}P_j E^k_{i,j} \end{bmatrix}. \tag{2.20}$$

P_j je polarizacija celice j, kjer vsota poteka po vseh celicah, ki so od celice i oddaljene manj od vnaprej definiranega efektivnega radija r_e. Z γ_i je označena vsota tunelirnih energij elektronov znotraj celic, ki so od celice i oddaljene manj kot r_e. Simulacije na osnovi bistabilnega modela se razlikujejo v metodah, s katerimi se na podlagi Hamiltonove matrike H_i za vsako celico i izračuna njena polarizacija P_i.

Bistabilna aproksimacija (angl. *bistable approximation*) [72, 37] je bistabilen model, ki temelji na nelinearni spremembi polarizacije QCA celice i glede na sosednje celice. Aproksimacija ne upošteva kvantnomehanske korelacije med celicami. Privzeto je adiabatno preklapljanje in sistem, ki se stalno nahaja zelo blizu osnovnega stanja. Z uporabo časovno neodvisne Schrödingerjeve enačbe se izračunajo rešitve izraza

$$H_i\psi_i = E_i\psi_i, \tag{2.21}$$

kjer je H_i Hamiltonova matrika, podana z enačbo (2.20), ψ_i vektor stanj celice i in E_i energija, povezana s stanji. Rešitve izraza (2.21) določajo polarizacijo P_i celice i

$$P_i = \frac{\frac{E^k_{i,j}}{2\gamma} \sum_j P_j}{\sqrt{1 + \frac{E^k_{i,j}}{2\gamma} \sum_j P_j}}, \tag{2.22}$$

kjer je γ tunelirna energija elektronov znotraj celice in $\sum_j P_j$ vsota polarizacij celic, ki so od celice i oddaljene manj od vnaprej definiranega efektivnega radija. Simulacija se izvede z iterativnim računanjem P_i za vsako celico v QCA, dokler je razlika med rezultatoma zaporednih iteracij večja od vnaprej predpisane tolerančne vrednosti ϵ. Z uporabo bistabilne aproksimacije je možna hitra simulacija delovanja QCA z velikim številom celic, ki pa ni primerna za simulacijo dinamike. Zato se ta model lahko uporablja predvsem za pogoste vmesne simulacije QCA med postopkom snovanja.

Za natančnejšo analizo je bolj primerna simulacija z uporabo *koherenčnega vektorja* (angl. *coherence vector*) [72]. Tudi ta je osnovana na bistabilnem modelu in za razliko od bistabilne aproksimacije upošteva časovno odvisnost in kvantnomehanske pojave. Koherenčni vektor λ je vektorska predstavitev gostotne matrike ρ, ki statistično določa kvantna stanja celice. Komponenta λ_i je podana kot sled matrike, ki je produkt gostotne matrike ρ in Paulijeve matrike σ_i. Izračuna se z izrazom

$$\lambda_i = tr(\rho\sigma_i), \quad i = \{x, y, z\}. \tag{2.23}$$

Polarizacija P_i je z komponenta λ_i, torej

$$P_i = \lambda_{z,i}. \tag{2.24}$$

Simulacija z uporabo koherenčnega vektorja je počasnejša od bistabilne aproksimacije, vendar bolj natančno obravnava dinamiko delovanja QCA.

2.8 Logični primitivi QCA

Procesiranje je v tem delu obravnavano kot računanje rezultata kompleksne logične funkcije. Vhodi in izhodi procesiranja ustrezajo vhodom in izhodom logične funkcije. Procesiranje je sestavljeno iz zaporedja logičnih operacij in povezav med njimi. Rezultati operacij v k-tem členu zaporedja se uporabijo kot vhodi operacij v členu $k + 1$. Zato so za procesiranje potrebni elementi, ki lahko izračunajo rezultate logičnih operacij in omogočajo prenos podatkov. *Logični primitivi* (angl. *logic primitives*) [73, 74] so realizacije takih elementov.

V QCA je logični primitiv razmestitev QCA celic na področjih v pravokotni mreži. Sestavljen je izključno iz QCA celic, ki so razdeljene na tri tipe, glede na vlogo v procesiranju in sicer na vhodne, notranje in izhodne celice. Število vhodov logičnega primitiva je enako številu vhodnih QCA celic n_{vhod} in podobno je število izhodov enako številu izhodnih QCA celic n_{izhod}.

Nosilec podatka je *signal*. Vhodni signali nosijo vhodne podatke, izhodni signali pa izhodne podatke, ki predstavljajo rezultat procesiranja. Procesiranje se prične s prihodom vhodnih signalov v vhodne celice in se nadaljuje s tokom in preoblikovanjem podatkov v notranjih celicah. Po zakasnitvi d, merjeni v številu pretečenih faz urinega signala, so izhodni signali na voljo v izhodnih celicah.

Logične vrednosti vhodnih in izhodnih signalov so elementi množice $B = \{0, 1\}$. Vsak signal s ima določeno natanko eno logično vrednost $v(s) \in B$. Delovanje logičnega primitiva je opisano s preslikavo $f : B^{n_{vhod}} \rightarrow B^{n_{izhod}}$, ki preslika logične vrednosti vhodnih signalov v logične vrednosti izhodnih signalov.

Logični primitivi v QCA so sestavljeni iz manjšega števila celic in izvajajo osnovne Boolove operacije, kot so na primer logična konjunkcija (AND), disjunkcija (OR), negacija (NOT), negacija disjunkcije (NOR), negacija konjunkcije (NAND), itd. Poleg realizacij osnovnih operacij omogočajo tudi prenos podatkov med njimi.

Poln funkcijski nabor je množica logičnih operacij, s katerimi je možno realizirati poljubno logično funkcijo. Med polne funkcijske nabore spadajo množice {AND, OR, NOT}, {NOR}, {NAND}, itd. Posebno pomembni so logični primitivi, ki realizirajo vse operacije v polnem funkcijskem naboru in povezave med njimi. Taki primitivi so potrebni in zadostni za realizacijo katerekoli logične funkcije. Za fizično implementacijo je zaželeno, da so primitivi relativno enostavni. Za QCA se je izkazalo, da je na tej osnovi možno sestaviti enostavne logične primitive, ki realizirajo navedene pomembne elemente.

2.8.1 Linija

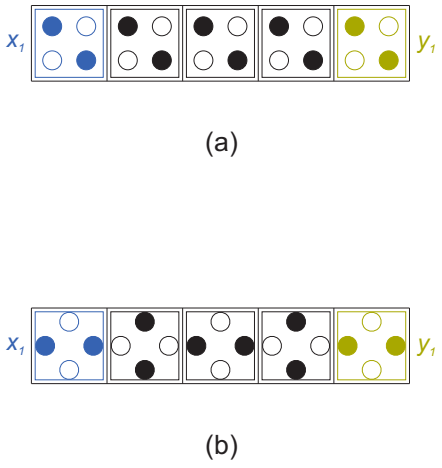
Prenos podatkov v QCA omogoča logični primitiv *linija* (angl. *QCA wire*). Ima en vhod, označen z x_1 in en izhod y_1. Linija skrbi za prenos signala od vhodne do izhodne celice, ne da bi se pri tem spremenila njegova logična vrednost. Zato izvaja logično operacijo identitete, torej $y_1 = x_1$. Realizirana je z namestitvijo QCA celic v središča zaporednih področij, kot je prikazano na sliki 2.13(a). Zaporedno nameščene QCA celice se polarizirajo z enako polarizacijo, kot jo ima vhodna celica. Tako se v izhodni celici

pojavi stanje, ki je enako stanju vhodne celice.

Linija je lahko sestavljena tudi iz rotiranih QCA celic. Takrat je logični primitiv poimenovan *45 stopinjska linija*. Taka linija še vedno izvaja logično operacijo identitete, če je število notranjih celic $n_{notranje}$ liho. V nasprotnem primeru je stanje izhodne celice enako negaciji stanja vhodne celice. V splošnem torej 45 stopinjska linija izvaja operacijo

$$y_1 = \begin{cases} x_1, & \text{če je } n_{notranje} = 2k+1, \\ \overline{x_1}, & \text{če je } n_{notranje} = 2k, \quad k \in \mathbb{N}. \end{cases} \tag{2.25}$$

Polarizacije celic v 45 stopinjski liniji alternirajo med $P = -1$ in $P = +1$, kot je prikazano v primeru linije na sliki 2.13(b).

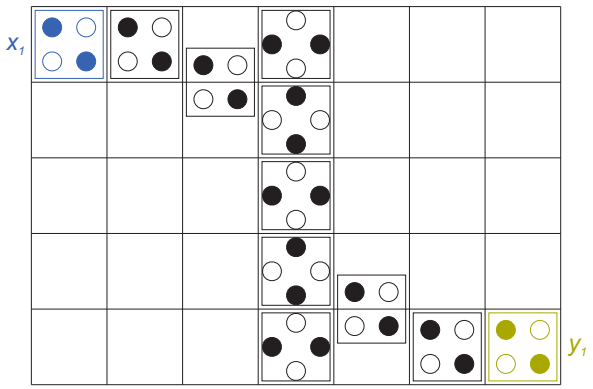


Slika 2.13 Liniji v QCA. Vhodna celica vsake linije je označena z x_1, izhodna celica pa z y_1. Slika (a) prikazuje linijo, sestavljeno iz navadnih QCA celic. Na sliki (b) je 45 stopinjska linija, sestavljena iz rotiranih QCA celic.

Celotna linija iz navadnih ali rotiranih QCA celic deluje pravilno, kadar je nameščena v eno samo urino cono, tako da je zakasnitev signala na njej $d = 1$. Pri tem pa je potrebno upoštevati maksimalno število QCA celic v liniji znotraj ene urine cone, podano z enačbo (2.15). Daljše linije je potrebno razdeliti v več urinih con.

Za prehod signala iz navadne na 45 stopinjsko linijo in obratno je potrebno QCA celico na stičišču linij zamakniti za polovico stranice področja navzgor ali navzdol. Središče te celice se namesto v središču področja nahaja na meji med dvema področjema. Slika 2.14 prikazuje dve navadni vodoravni liniji in eno navpično 45 stopinjsko, ki se med seboj stikajo. Signal potuje od vhodne celice x_1 v zgornjem levem kotu mreže do izhodne celice y_1 v spodnjem desnem kotu. QCA celica na zgornjem stičišču linij je zamaknjena

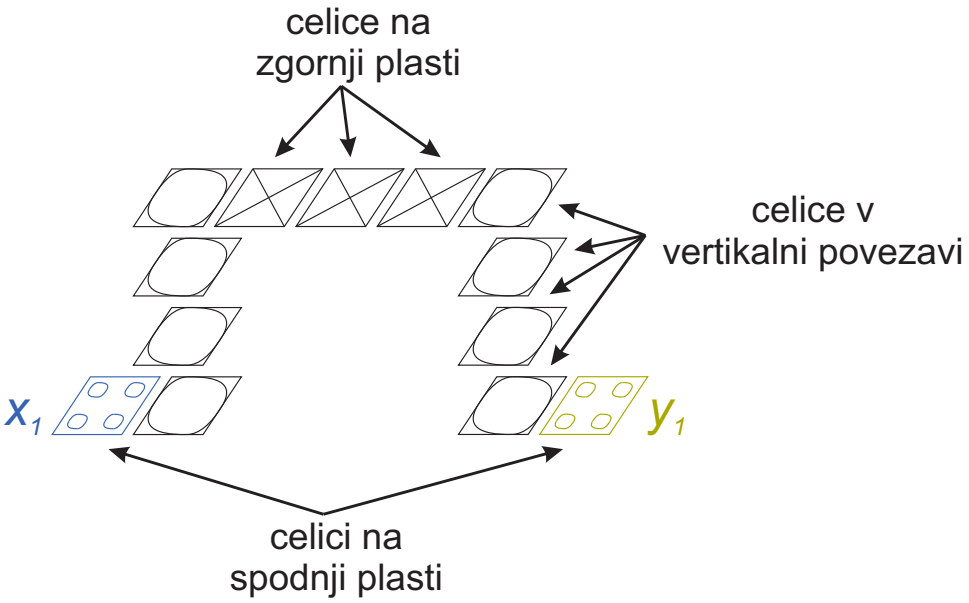
za $\frac{c}{2}$ navzdol, kjer je c dolžina stranice področja v mreži. Logična vrednost signala se ob prehodu iz navadne linije ohrani in nato alternira med potekom po 45 stopinjski liniji navzdol. Tudi prenos signala iz 45 stopinjske na navadno linijo je realiziran z zamaknjeno celico, pri čemer se njegova logična vrednost ne spremeni. Na sliki 2.14 je QCA celica na stičišču navpične 45 stopinjske in spodnje vodoravne linije zamaknjena za $\frac{c}{2}$ navzgor. Stanje izhoda y_1 je odvisno od števila QCA celic v 45 stopinjski liniji, kot je to določeno z izrazom (2.25). Vse celice na sliki 2.14 se nahajajo v isti urini coni.



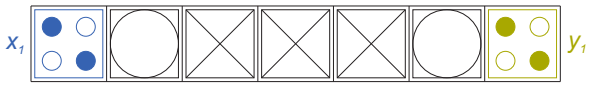
Slika 2.14 Dve navadni vodoravni in ena navpična 45 stopinjska linija. Signal poteka od vhodne celice x_1 po vodoravni liniji proti desni, nato preide na navpično 45 stopinjsko linijo in poteka po njej navzdol, spodaj pa preide na navadno vodoravno linijo in poteka po njej proti desni do izhodne celice y_1. Celica na zgornjem stičišču linij je zamaknjena za $\frac{c}{2}$ navzdol, celica na spodnjem stičišču pa za $\frac{c}{2}$ navzgor.

Linija lahko prehaja iz spodnje na zgornjo plast in obratno preko vertikalne povezave. Slednja poteka od spodnje plasti preko dveh plasti do zgornje, kot je prikazano na sliki 2.15. S tem je onemogočena neželena interakcija med celicami na spodnji plasti in celicami, ki ležijo na na zgornji plasti. Linija iz slike 2.15 je dvodimenzionalno prikazana na sliki 2.16.

V kompleksnejših procesnih sistemih se pogosto pojavi potreba po mehanizmu, ki omogoča spremembo smeri poteka signala. V primeru prehoda iz navadne vodoravne na navpično 45 stopinjsko linijo se smer poteka obrne za 90°, vendar nato signal poteka po rotiranih QCA celicah. V QCA je možno spremeniti smer poteka signala po navadnih celicah s t.i. *kotno linijo*, prikazano na sliki 2.17. Tako kot navadna, ima tudi kotna linija en vhod x_1, en izhod y_1 in izvaja logično operacijo identitete $y_1 = x_1$. Realizacija kotne linije na sliki 2.17(a) spremeni smer poteka signala za 90° v smeri urinega kazalca, realizacija na sliki 2.17(b) pa za 90° v nasprotni smeri. S stikanjem vodoravnih, navpičnih



Slika 2.15 Linija preide iz spodnje na zgornjo plast, nato poteka po zgornji plasti in nazadnje spet preide na spodnjo plast.

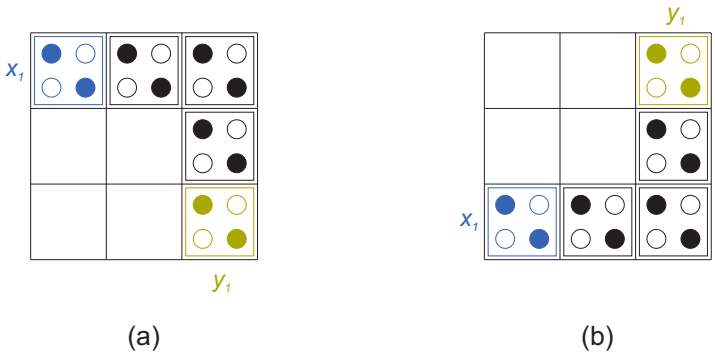


Slika 2.16 Dvodimenzionalni prikaz linije iz slike 2.15.

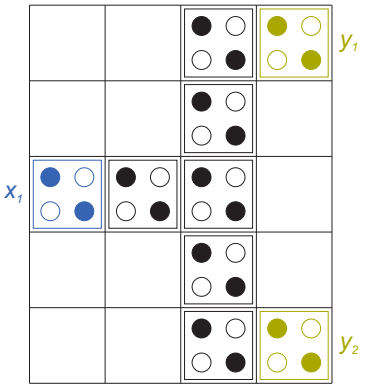
in kotnih linij je možno prenesti signal od vhodne celice do celice, ki se nahaja na poljubno izbranem področju v mreži. Kotna linija lahko vsebuje več kotov, za njeno delovanje pa je potrebna ena urina faza.

Pri izračunavanju nekaterih logičnih funkcij je potrebno isti signal pripeljati na več vhodov. Primer takšne funkcije je $f(X_1, X_2) = X_1 \uparrow (X_1 \uparrow X_2)$, kjer mora signal X_1 priti na vhoda dveh logičnih primitivov, ki realizirata operacijo NAND ($\uparrow$). *Razvejitev signala* (*angl. fan-out*) je v QCA omogočeno z *razvejitveno linijo*, ki je prikazana na sliki 2.18. Signal na vhodu x_1 se v razvejitveni liniji podvoji, tako da sta na obeh izhodih y_1 in y_2 prisotna signala z enako logično vrednostjo, kot jo ima vhodni signal. Logični primitiv torej izvaja operacijo $y_1 = y_2 = x_1$. Razvejitev signala se izvede v eni urini fazi.

V kompleksnem logičnem vezju se nahaja veliko število povezav, ki prenašajo signale med logičnimi primitivi. Zaradi tega se morajo povezave med seboj večkrat križati. Za snovanje in izdelavo kompleksnega QCA je torej potrebno vzpostaviti mehanizem, ki



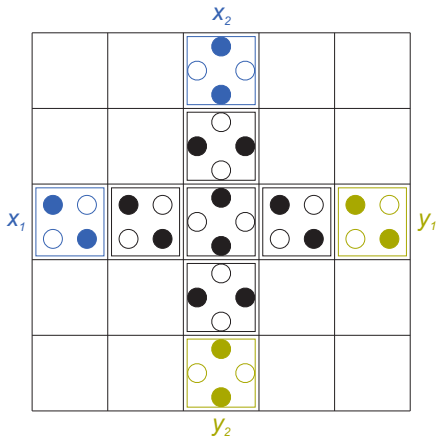
Slika 2.17 Logični primitiv kotna linija omogoča spremembo smeri poteka signala za 90°. Realizacija na sliki (a) spremeni potek v smeri urinega kazalca, realizacija na sliki (b) pa v nasprotni smeri.



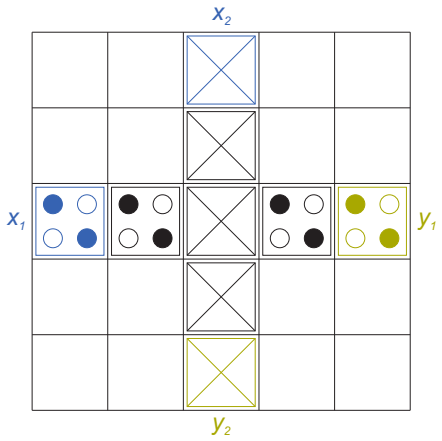
Slika 2.18 Razvejitvena linija.

omogoča križanje. Logični primitiv, ki omogoča križanje dveh QCA linij, je predstavil Tougaw [4]. Njegova zanimiva lastnost je, da se liniji križata v isti ravnini. Tougaw je pokazal, da se vhodna signala x_1 in x_2 ohranita na navadni in na 45 stopinjski liniji tudi po križanju, kot je prikazano na sliki 2.19. Za izhodna signala veljata enačbi $y_1 = x_1$ in $y_2 = x_2$. Na točki križanja vodoravne navadne in navpične 45 stopinjske linije je nameščena rotirana QCA celica, kljub temu pa se signal nemoteno prenaša tudi po navadni liniji. Zakasnitev signala v predstavljenem logičnem primitivu je ena urina faza.

Večnivojsko križanje linij je izvedeno tako, da linija na zgornji plast poteka nad linijo na spodnji plasti. Na točki križanja se nahaja ena celica na spodnji in ena na zgornji plasti. Ker sta med spodnjo in zgornjo plastjo še dve vmesni plasti, med tema celicama ni neželene interakcije. Večnivojsko križanje je prikazano na sliki 2.20.



Slika 2.19 Križanje vodoravne navadne in navpične 45 stopinjske linije v isti ravnini.



Slika 2.20 Križanje linije na spodnji in linije na zgornji plasti.

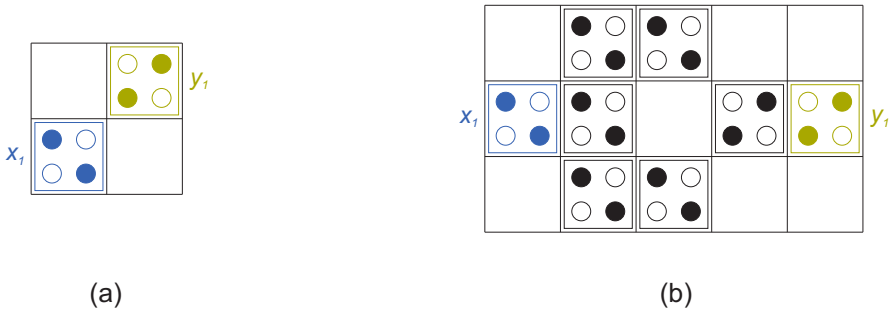
2.8.2 Negator

Pogosto se pri procesiranju pojavlja operacija negacije logične vrednosti vhodnega signala. Realizirana je z *negatorjem*, ki ima en vhod in en izhod. Negacija je pomembna tudi zaradi tega, ker je del polnega funkcijskega nabora {AND, OR, NOT}. V QCA je lahko negator realiziran na več različnih načinov. Ena od možnosti je 45 stopinjska linija s sodim številom notranjih celic, kot je razvidno iz izraza (2.25). Ta rešitev ni praktična takrat, ko se signal že nahaja na navadni liniji, saj je v tem primeru potrebno poskrbeti za prenos signala med navadno in 45 stopinjsko linijo.

V QCA je možno realizirati osnovni negator z enostavno diagonalno razporeditvijo

dveh QCA celic, prikazano na sliki 2.21(a). Takšna razporeditev je geometrijsko ekvivalentna diagonalno postavljeni 45 stopinjski liniji, sestavljeni iz dveh rotiranih QCA celic. Osnovni negator izvede logično operacijo $y_1 = \overline{x_1}$ v eni urini fazi.

Glede na izračune uporaba osnovnega negatorja poveča možnost, da QCA med procesiranjem preide v nezaželeno metastabilno stanje [70]. To težavo je možno odpraviti z razdelitvijo logičnega primitiva na sliki 2.21(a) v dve urini coni, tako da x_1 pripada prvi in y_1 drugi coni. Tedaj je zakasnitev signala pri negaciji enaka dvema urinima fazama, kar pa ni zaželeno za tako pogosto in enostavno operacijo. Izkazalo se je [4, 75], da lahko v QCA operacijo logične negacije $y_1 = \overline{x_1}$ realizira robusten negator z zakasnitvijo $d = 1$, prikazan na sliki 2.21(b). Signal poteka od vhodne celice x_1, se nato podvoji in



Slika 2.21 Realizacija osnovnega negatorja z dvema QCA celicama (a) in realizacija robustnega negatorja (b).

preide na zgornjo in spodnjo vodoravno linijo. Na desni strani se izvede negacija vrednosti signala z diagonalno nameščeno QCA celico med zgornjo in spodnjo linijo. V tem primeru je zaradi dveh linij negacija izvedena bolj robustno kot pri osnovnem negatorju. Prednost robustnega negatorja je tudi v tem, da se vhodna in izhodna celica nahajata na isti vodoravni premici. Tako lahko signal poteka po ravni vodoravni liniji, v nasprotju z osnovnim negatorjem, kjer preide na za eno področje višje nameščeno linijo. Očitna slabost robustne izvedbe v primerjavi z osnovno je večje število uporabljenih QCA celic in večja zasedena površina v mreži.

2.8.3 Majoritetna vrata

Logični primitiv *majoritetna vrata* (angl. *majority gate*) s tremi vhodi x_1, x_2, x_3 in izhodom y_1 izvaja *majoritetno* ali *večinsko* funkcijo

$$y_1 = M(x_1, x_2, x_3) = x_1x_2 \vee x_2x_3 \vee x_1x_3. \tag{2.26}$$

Logična vrednost izhoda y_1 je enaka vrednosti, ki jo ima večina vhodov. Majoritetna funkcija je komutativna, kar pomeni, da vrednost izhoda ni odvisna od vrstnega reda vhodov. Če $\pi_i(x_1, x_2, x_3)$ označuje permutacijo z indeksom i v množici permutacij treh spremenljivk x_1, x_2, x_3, potem za poljubna indeksa i in j velja

$$M(\pi_i(x_1, x_2, x_3)) = M(\pi_j(x_1, x_2, x_3)), \quad 1 \leq i, j \leq 3!. \tag{2.27}$$

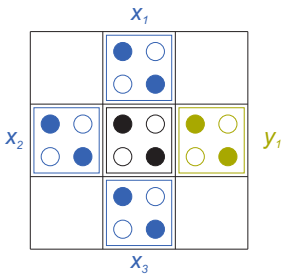
Pomembna lastnost majoritetnih vrat je, da je z njimi mogoče realizirati tako logično konjunkcijo, kot tudi logično disjunkcijo. Če ima na primer vhod x_3 fiksno logično vrednost 0, se enačba (2.26) poenostavi v

$$y_1 = M(x_1, x_2, 0) = x_1 x_2, \tag{2.28}$$

tako da je y_1 rezultat konjunkcije vhodov x_1 in x_2. Zaradi komutativnosti je lahko vrednost kateregakoli vhoda fiksirana na logično 0 in takrat je rezultat majoritetne funkcije logična konjunkcija preostalih dveh vhodov. Kadar je vrednost enega izmed vhodov fiksirana na logično 1, je rezultat majoritetne funkcije logična disjunkcija preostalih dveh vhodov. V primeru fiksne vrednosti $x_3 = 1$, je enačba majoritetne funkcije

$$y_1 = M(x_1, x_2, 1) = x_1 \vee x_2. \tag{2.29}$$

Realizacijo majoritetnih vrat v QCA sestavlja pet QCA celic, razmeščenih v obliki križa. Prikazana je na sliki 2.22(a), na sliki 2.22(b) pa je pravilnostna tabela majoritetne funkcije. Vse celice so nameščene v isti urini coni, tako da je zakasnitev signala v



(a)

x_1	x_2	x_3	y_1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b)

Slika 2.22 Majoritetna vrata (a) in pravilnostna tabela majoritetne funkcije (b).

majoritetnih vratih $d = 1$.

Iz opisa logičnih primitivov QCA v tem razdelku je razvidno, da negator in majoritetna vrata realizirata vse operacije v polnemu funkcijskemu naboru {AND, OR, NOT}. S predstavljenimi linijami je omogočena poljubna povezava njunih vhodov in izhodov. Iz tega sledi, da je z uporabo negatorjev, majoritetnih vrat in linij teoretično mogoče realizirati poljubno logično funkcijo. Našteti primitivi so torej zadostni za sestavo procesorja na osnovi QCA.

2.9 Logične strukture QCA

Poljubna *logična funkcija* z n_{vhod} vhodi in n_{izhod} izhodi je opisana s preslikavo

$$f : B^{n_{vhod}} \rightarrow B^{n_{izhod}}, \tag{2.30}$$

kjer je $B = \{0, 1\}$. Sestavljena je iz logičnih operacij, s katerimi se vrednosti vhodov preslikajo v vrednosti izhodov. Za primer vzemimo funkcijo polovičnega seštevalnika $add(A, B) = (S, C)$ z vhodoma A, B ($n_{vhod} = 2$) in izhodoma S, C ($n_{izhod} = 2$). Vhoda določata vhodna bita, ki ju je potrebno sešteti, izhod S je vsota vhodov po modulu 2, C pa določa prenos. Logično funkcijo opišemo tako, da podamo logične izraze za izračun vseh izhodov. Tako lahko funkcijo $add(A, B) = (S, C)$ opišemo z izrazoma

$$\begin{aligned} S &= A\overline{B} \vee \overline{A}B, \\ C &= AB. \end{aligned} \tag{2.31}$$

Izraza (2.31) vsebujeta le operacije iz nabora {AND, OR, NOT}, lahko pa bi izbrali tudi kakšen drug funkcijsko poln nabor operacij. V primeru, da uporabljamo le operacijo NAND, lahko funkcijo $add(A, B) = (S, C)$ zapišemo kot

$$\begin{aligned} S &= ((A \uparrow A) \uparrow B) \uparrow ((B \uparrow B) \uparrow A), \\ C &= (A \uparrow B) \uparrow (A \uparrow B). \end{aligned} \tag{2.32}$$

Opis funkcije polovičnega seštevalnika z izrazoma (2.31) vsebuje 3 operacije konjunkcije, dve negaciji in eno disjunkcijo, opis (2.32) pa zahteva osem operacij NAND.

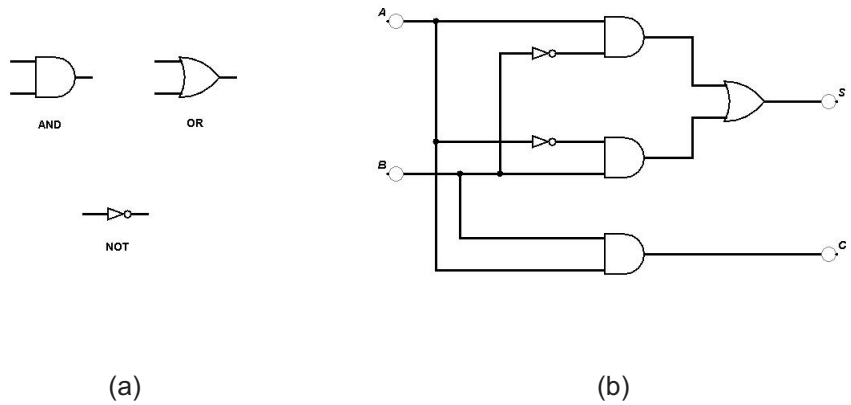
Logično funkcijo lahko torej izrazimo na več načinov z uporabo različnih funkcijsko polnih naborov operacij. Za enolični zapis funkcij se uporabljajo normalne oblike, kot je na primer popolna disjunktivna normalna oblika. Funkcija se pred implementacijo

običajno minimizira, tako da jo sestavlja čim manjše število operacij iz danega nabora. Minimizacija se izvede v postopku logičnega snovanja, ki ga v tem delu ne bomo obravnavali. Enoličen zapis logične funkcije je možen tudi z uporabo pravilnostne tabele. Slednja je za funkcijo polovičnega seštevalnika $add(A, B) = (S, C)$ predstavljena s tabelo 2.1.

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

Tabela 2.1 Pravilnostna tabela polovičnega seštevalnika $add(A, B) = (S, C)$.

Logična shema je grafičen prikaz funkcije, podane z logičnimi izrazi. Ker lahko isto funkcijo opisujejo različni izrazi, logična shema ni enolična predstavitev logične funkcije. Sestavljajo jo vhodni in izhodni priključki ter medsebojno povezani logični operatorji. Vhodni priključki ustrezajo vhodom funkcije, izhodni priključki izhodom in operatorji logičnim operacijam. V logični shemi so operatorji prikazani z dogovorjenimi grafičnimi simboli. Na sliki 2.23(a) so prikazani simboli operatorjev, ki predstavljajo logične operacije iz nabora {AND, OR, NOT}. Slika 2.23(b) prikazuje logično shemo polovičnega seštevalnika, opisanega z izrazoma (2.31).

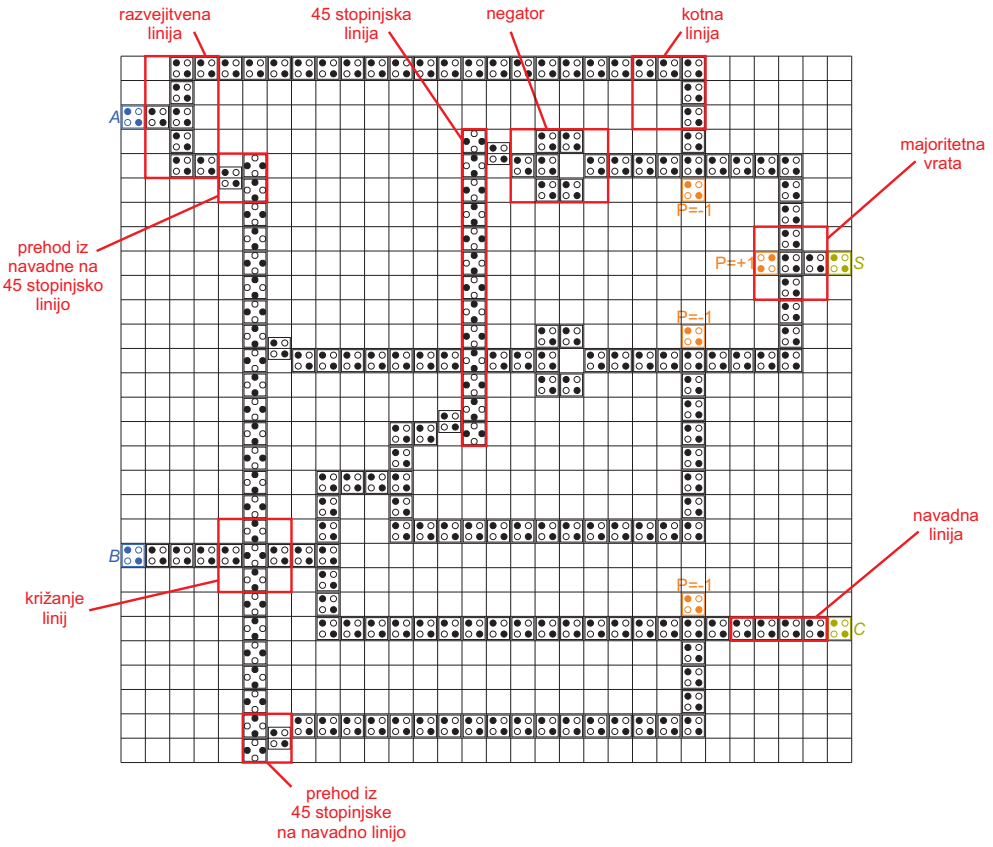


Slika 2.23 Grafični simboli operatorjev, ki predstavljajo logične operacije AND, OR in NOT (a). Logična shema polovičnega seštevalnika $add(A, B) = (S, C)$ z vhodnima priključkoma A, B in izhodnima priključkoma S, C (b).

Logična shema nastane v postopku logičnega snovanja, ki ga v tem delu ne bomo obravnavali. Logične sheme zato ne bomo izdelovali, ampak bomo privzeli, da je že podana.

Logična struktura (angl. *logic structure*) [76, 52] je realizacija dane logične sheme in s tem tudi logične funkcije. Ker lahko funkcijo opisuje več različnih shem, zanjo obstaja tudi več logičnih struktur, določena struktura pa realizira natanko eno funkcijo. Logično strukturo sestavljajo logični primitivi, ki realizirajo operacije in povezave med njimi glede na shemo. Ker so vsi primitivi v QCA zgrajeni s QCA celicami, je tudi logična struktura QCA sestavljena izključno iz celic.

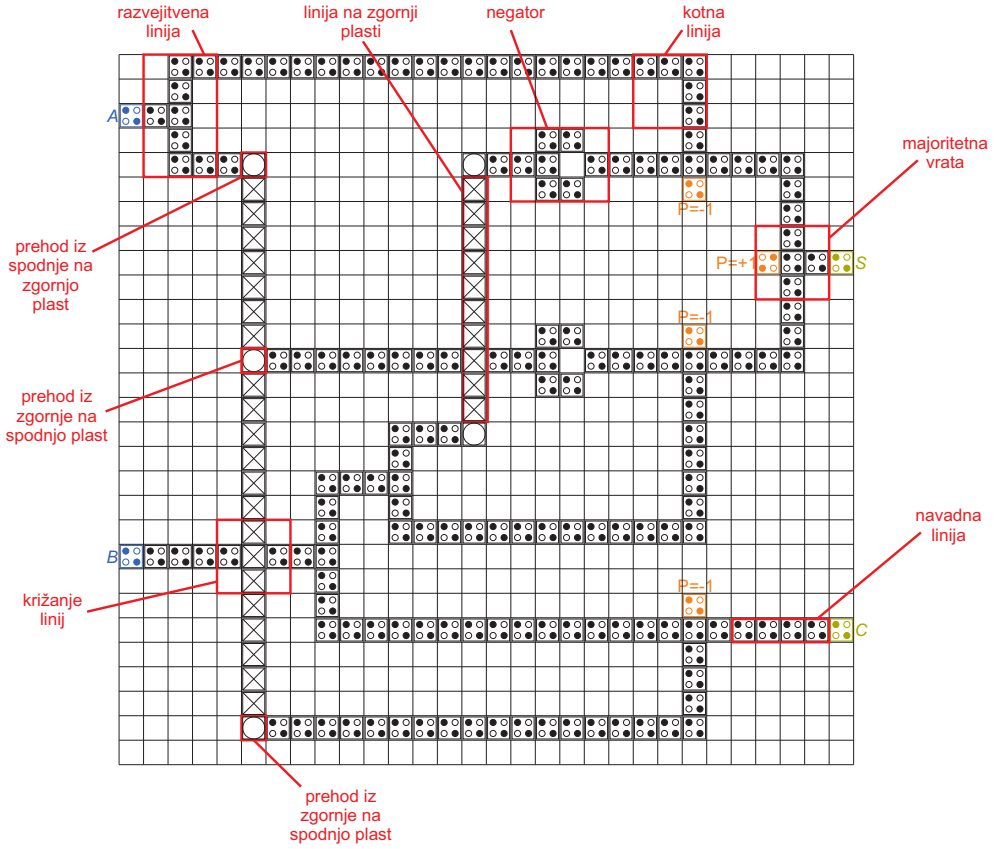
Z ad hoc postopkom se lahko zasnuje logična struktura tako, da se elementi v logični shemi neposredno realizirajo z logičnimi primitivi. Tako je zasnovana struktura QCA z označenimi primitivi na sliki 2.24. Prikazana struktura je neposredna realizacija logične



Slika 2.24 Ad hoc zasnovana logična struktura polovičnega seštevalnika $add(A, B) = (S, C)$ z vhodnima celicama A, B in izhodnima celicama S, C. Označeni so nekateri logični primitivi.

sheme s slike 2.23(b) na osnovi QCA. Celice s fiksno polarizacijo določajo logično operacijo, ki jo izvajajo majoritetna vrata. Pri polarizaciji $P = -1$ izvajajo operacijo AND, pri $P = +1$ pa operacijo OR.

Strukturo je možno realizirati tudi z uporabo večnivojskega križanja linij. V tem primeru se namesto 45 stopinjske linije uporabi linijo na zgornji plasti, namesto prehoda iz navadne na 45 stopinjsko linijo pa se namesti vertikalna povezava. Realizacija strukture iz slike 2.24 z uporabo večnivojskega križanja linij je prikazana na sliki 2.25.



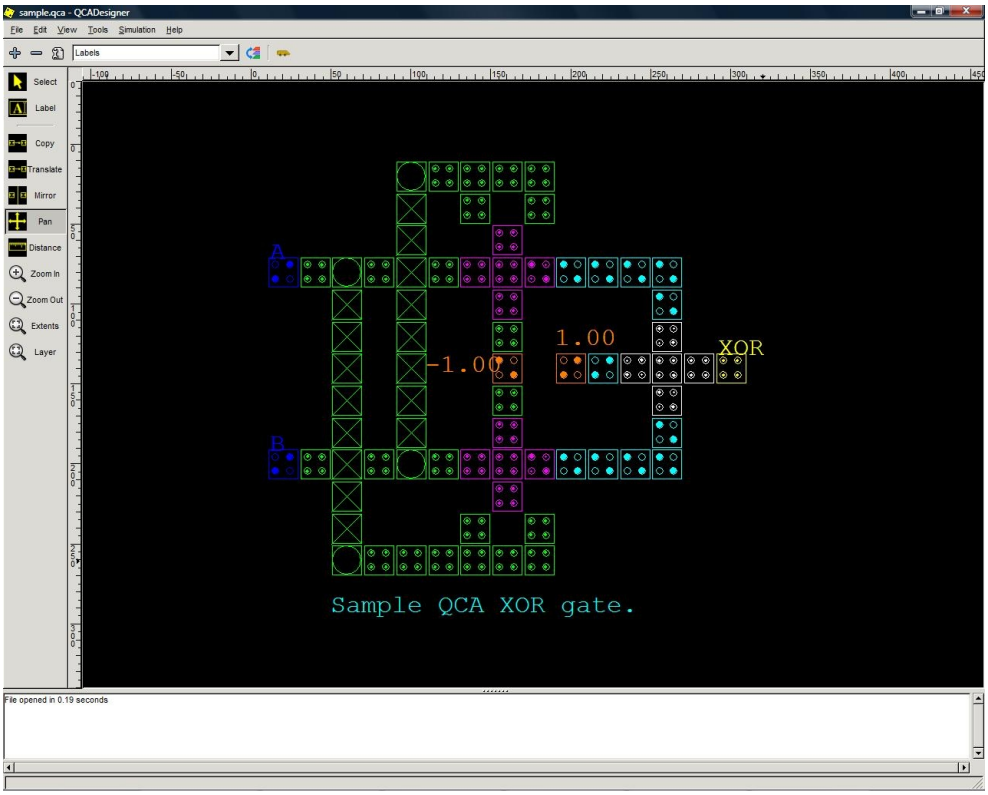
Slika 2.25 Struktura iz slike 2.24, pri kateri je namesto križanja linij v isti ravnini uporabljeno večnivojsko križanje.

2.10 Načrtovalsko programsko orodje QCADesigner

QCADesigner je orodje, namenjeno za računalniško podporo pri snovanju logičnih struktur na osnovi QCA celic. Razvil ga je Walus s sodelavci [52, 37, 75] leta 2003. Programska

koda orodja je prosto dostopna na svetovnem spletu[1]. QCADesigner omogoča vizualno postavitev in simulacijo logičnih struktur. V tem delu ga bomo uporabljali kot orodje za simulacijo in verifikacijo pravilnosti delovanja zasnovanih logičnih struktur.

Enostavno delo z orodjem omogoča grafični uporabniški vmesnik, prikazan na sliki 2.26. Sestavljajo ga menijska vrstica, orodni vrstici na zgornjem in levem robu, okno za prikaz informacij na spodnjem robu in delovna površina na sredini. Uporaba ikon v orodnih vrsticah omogoča vizualno nameščanje QCA celic na delovno površino in manipuliranje z njimi. Podprto je nameščanje navadnih in rotiranih celic in njihovo premikanje po površini. Pri tem orodje opozarja na nedovoljene situacije, kot je na primer prekrivanje dveh QCA celic na isti plasti.



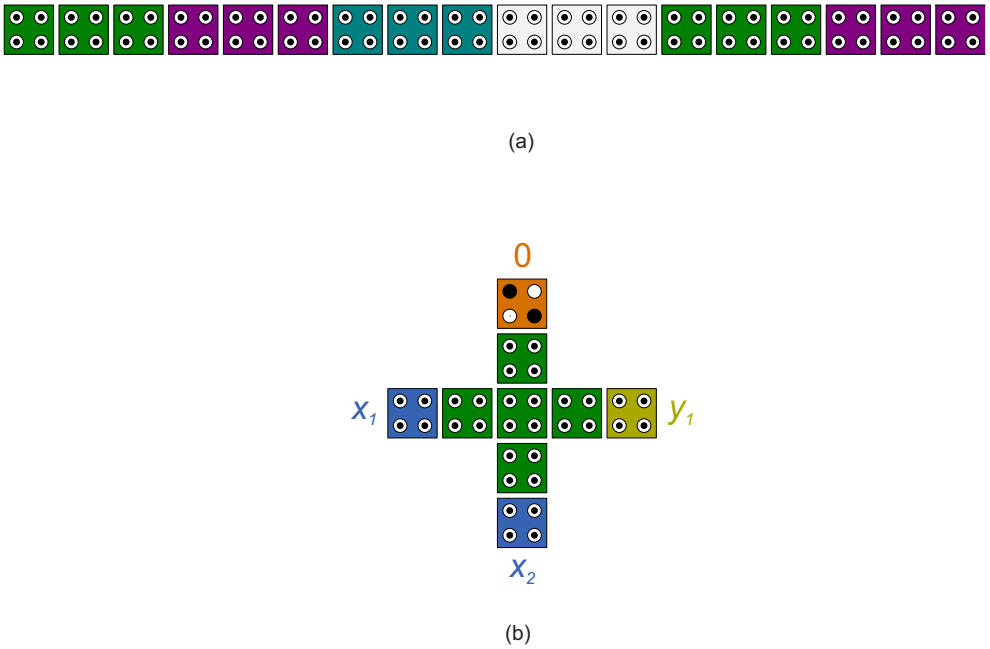
Slika 2.26 Grafični uporabniški vmesnik računalniškega orodja QCADesigner.

Za vse celice skupaj se pred nameščanjem določi njihova velikost in premer kvantnih pik. Ko je celica nameščena na površino, se ji predpiše ena izmed štirih vlog v logični

[1] http://www.mina.ubc.ca/qcadesigner

strukturi. Celica je lahko vhodna, izhodna, lahko ima fiksno polarizacijo ali pa nima posebne vloge. Vloga celice se v vmesniku kaže z njeno barvo in oznako.

Vsaki celici posebej ali pa celotni skupini celic se določi pripadajoča urina cona. Tudi pripadnost urini coni se kaže z barvo celice, če slednja nima posebne vloge. Barva vhodnih, izhodnih in celic s fiksno polarizacijo je nespremenljiva. Razen prej naštetih so celice, ki jih kontrolira isti urin signal, enako obarvane. Vse celice v isti urini coni imajo enako barvo, poleg tega pa imajo enako barvo tudi vse celice v drugih urinih conah z enako urino fazo. Tako so za označevanje urinih con namenjene štiri različne barve, kot je prikazano na sliki 2.27(a). Prikazana vodoravna linija je razdeljena na šest urinih con,

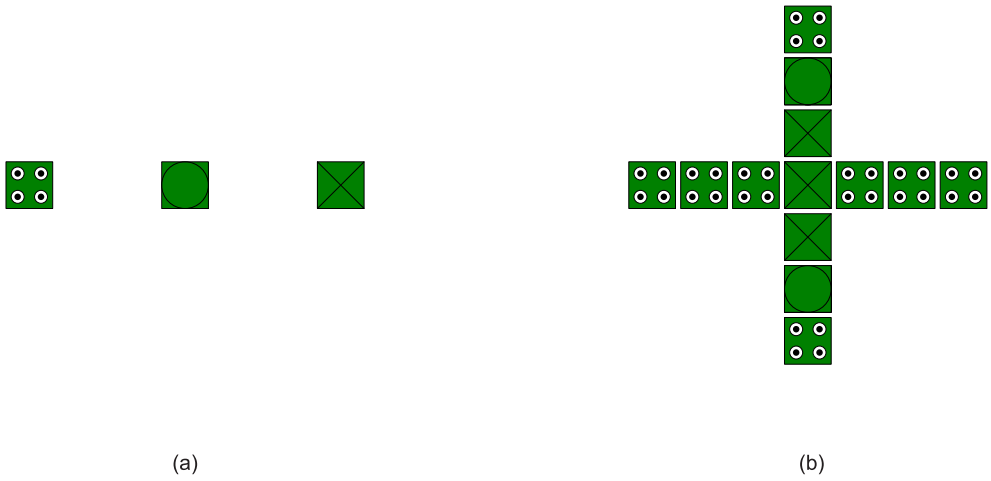


Slika 2.27 Vodoravna linija, razdeljena v šest urinih con. Celice z enako barvo kontrolira isti urin signal (a). Logična struktura, ki jo sestavljajo vhodni celici x_1 in x_2, celica s fiksno polarizacijo -1 (logična vrednost 0), majoritetna vrata in izhodna celica y_1. Vsaka celica je obarvana glede na njeno vlogo v strukturi (b).

ki si zaporedno sledijo od leve proti desni. Vsaka urina cona vsebuje tri celice. V conah z zelenimi celicami je urin signal na začetku procesiranja v fazi preklopa. V zaporedju si nato sledijo cona z vijoličnimi celicami v fazi sproščenosti, cona s sinjimi celicami v fazi sproščanja in cona z belimi celicami v fazi zadrževanja, nato pa se barve ciklično ponavljajo. Slika 2.27(b) prikazuje logično strukturo, ki jo sestavljajo dve vhodni celici, ena izhodna, ena celica s fiksno polarizacijo in celice v majoritetnih vratih. Slednje se

nahajajo v isti urini coni in so obarvane z zeleno barvo. Vhodni celici sta modri, izhodna rumena, celica s fiksno polarizacijo pa je oranžne barve.

QCADesigner omogoča gradnjo večnivojskih logičnih struktur. Nad prvo plastjo substrata s QCA celicami se lahko namesti poljubno število plasti z določeno medsebojno razdaljo. Dodatne plasti se običajno uporabljajo za implementacijo večnivojskega križanja linij. Na prvi plasti se nahaja logična struktura brez križanj linij. Na zgornjih dveh plasteh so nameščene celice v vertikalnih povezavah med prvo in četrto plastjo, ki se nahaja najvišje. Slednja je namenjena za namestitev linij, ki se križajo z linijami na prvi plasti. Grafični vmesnik orodja QCADesigner omogoča tri različne vizualne prikaze celic, ki se uporabljajo za prikaz celic na različnih plasteh. Predstavljeni so na sliki 2.28(a). Križanje vodoravne linije na prvi plasti in navpične na četrti je prikazano na sliki 2.28(b).



Slika 2.28 Trije različni vizualni prikazi celic. Prvi z leve se uporablja za prikaz celic na prvi plasti, srednji za prikaz celic v vertikalni povezavi in desni za prikaz celic na četrti plasti (a). Večnivojsko križanje linij. Vodoravna linija se v celoti nahaja na prvi plasti. Navpična linija preide iz prve plasti po vertikalni povezavi do četrte plasti. Po njej poteka nad vodoravno linijo in nato po vertikalni povezavi preide nazaj na prvo plast (b).

Popoln opis logične strukture lahko orodje zapiše v strukturiranem formatu in ga shrani v tekstovno datoteko. V njej so dostopni tudi nekateri parametri, ki jih v grafičnem uporabniškem okolju ni mogoče spreminjati. Tako je možno zgraditi logično strukturo v orodju QCADesigner tudi z zapisovanjem parametrov v tekstovno datoteko.

Poleg vizualnega snovanja logične strukture je z orodjem možna tudi simulacija njenega delovanja. QCADesigner ponuja dva načina simulacije in sicer z uporabo bistabilne aproksimacije in z uporabo koherenčnega vektorja. Pred izvedbo simulacije se določijo

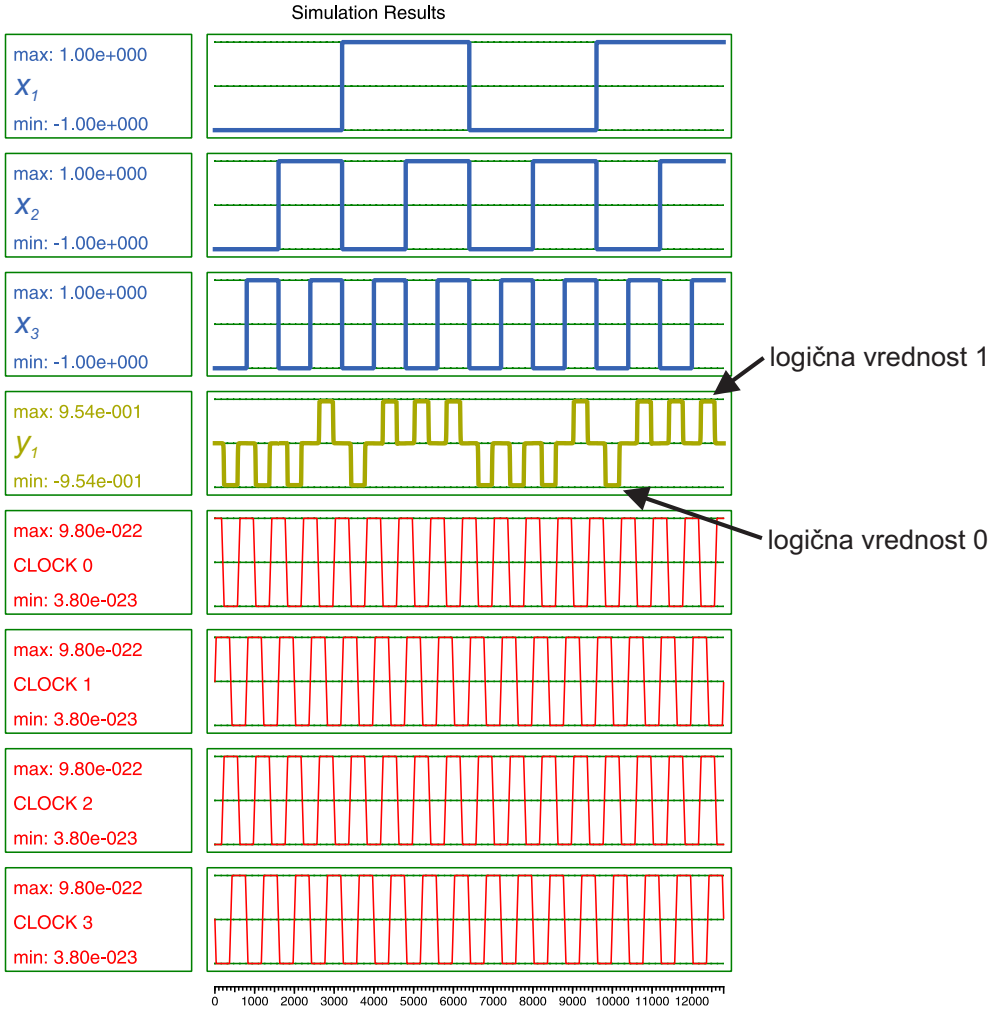
njeni parametri, kot so efektivni radij, relativna dielektričnost medija, itd. Po končani simulaciji se rezultati grafično predstavijo s prikazom na sliki 2.29. Predstavljeni so rezultati simulacije delovanja majoritetnih vrat z vhodi x_1, x_2, x_3 in izhodom y_1. Vse celice v majoritetnih vratih se nahajajo v eni urini coni. Grafi rdeče barve prikazujejo potek štirih urinih signalov skozi čas. Urini signali so poimenovani CLOCK 0, CLOCK 1, CLOCK 2 in CLOCK 3. Visoko stanje urinega signala ustreza fazi sproščenosti, nizko stanje pa fazi zadrževanja. Rumeni graf kaže spremembo polarizacije izhodne celice v odvisnosti od časa. Zanima nas polarizacija v časovnih intervalih, ko izhodno celico nadzoruje urin signal v fazi zadrževanja, torej kadar je v nizkem stanju. Polarizacija je takrat enaka -1 ali +1 in jo lahko interpretiramo kot logično vrednost. Izhodno celico y_1 nadzoruje signal CLOCK 0, zato nas zanima njena polarizacija takrat, ko je CLOCK 0 v nizkem stanju. Spreminjanje polarizacij vhodnih celic prikazujejo modri grafi. Te polarizacije alternirajo med -1 in +1, kar ustreza logičnima vrednostma 0 in 1. Tako tvorijo vse možne kombinacije vrednosti vhodov. Kot je razvidno s slike 2.29, se simulacija izvede dvakrat za vsako kombinacijo vhodnih vrednosti. Polarizacija izhodne celice je odvisna od kombinacije polarizacij vhodnih celic. Polarizacije vhodov in izhodov lahko zapišemo z logičnimi vrednostmi in z njimi sestavimo tabelo, ki natanko ustreza pravilnostni tabeli majoritetne funkcije.

2.11 Raziskave na področju QCA

2.11.1 Zgodovina raziskav

Kvantni celični avtomat je predstavil Lent s sodelavci leta 1993 [3]. Predstavili so QCA celico, ki je osnovni gradnik kvantnega celičnega avtomata in prikazali možnost uporabe celic za procesiranje. Delovanje QCA temelji na principih procesiranja z osnovnim stanjem in *robno gnanega procesiranja* (angl. *edge-driven computing*). Slednje omejuje interakcijo med QCA in okoljem, ki lahko poteka le preko vhodnih in izhodnih celic. Princip procesiranja z osnovnim stanjem določa, da je za procesiranje sprejemljivo le stanje QCA z najnižjim energijskim nivojem, imenovano osnovno stanje. Avtorji so pokazali tudi možnost realizacije QCA celice.

Tougaw in Lent [4] sta zasnovala QCA linijo, negator in majoritetna vrata. To je omogočilo izvedbo logičnih operacij negacije, konjunkcije in disjunkcije ter prenosa podatkov po liniji. Tako se je odprla pot za snovanje logičnih struktur QCA na podlagi



Slika 2.29 Grafični prikaz rezultatov simulacije delovanja majoritetnih vrat z vhodi x_1, x_2, x_3 in izhodom y_1. Visoko stanje vhodnega ali izhodnega signala ustreza logični vrednosti 1, nizko stanje signala pa ustreza logični vrednosti 0.

logičnih shem.

Pri gradnji večjih struktur so naleteli na problem metastabilnih stanj. Procesiranje se je ustavilo v lokalnem energijskem minimumu namesto v osnovnem stanju. Metastabilna stanja niso zaželena, saj iz njih ni mogoče razbrati uporabne logične vrednosti. Dosedanje grobo preklapljanje je nadomestilo adiabatno preklapljanje, omogočeno z uvedbo ure za nadzor preklopa [27].

Za simuliranje delovanja logičnih struktur QCA so uporabili več različnih simulacijskih modelov. Eden najpreprostejših je časovno neodvisni polklasični model [71]. Ker ta ne upošteva adiabatnosti, se za simuliranje QCA uporabljajo drugi modeli, med katerimi sta posplošeni Hubbardov model [3] in medcelična Hartreejeva aproksimacija [77]. Raziskovali so tudi statistične modele, kot sta uporaba Monte Carlo simulacije [78] in Bayesovih mrež [79, 80].

Pojavili so se predlogi za različne realizacije QCA celice. Najprej so bili predstavljeni polprevodniški QCA [3], sledili so jim prevodniški QCA s kvantnimi pikami na osnovi kovinskih otokov [56, 57], molekularni [35, 66] in nazadnje še magnetni QCA [67].

Niemier in sodelavci [19, 28, 81, 82, 20] so leta 2000 pričeli raziskovati snovanje kompleksnejših logičnih struktur. V svojem doktoratu [20] je Niemier določil nekatera načrtovalska pravila, ki morajo biti upoštevana pri snovanju.

Za snovanje kompleksnejših logičnih struktur je bilo potrebno razviti računalniško orodje z grafičnim uporabniškim vmesnikom. Tako je leta 2003 Walus s sodelavci razvil orodje QCADesigner [52, 37], ki snovalcem omogoča vizualno postavljanje logičnih struktur QCA in njihovo simulacijo.

V zadnjem času so avtorji podali različne predloge za snovanje logičnih struktur QCA [83, 5, 6]. Za formalizacijo procesa snovanja logičnih struktur QCA so poskušali uporabiti znanje s področja snovanja CMOS integriranih vezij [34]. Začeli so reševati probleme s področja razmeščanja in povezovanja v QCA [84, 21, 31, 30, 85, 86, 87, 32, 23, 88, 24, 89, 33].

2.11.2 Raziskave na področju razmeščanja in povezovanja v QCA

S snovanjem kompleksnih logičnih struktur QCA se je leta 2000 začel ukvarjati Niemier s sodelavci [19]. Zasnovali so enostaven mikroprocesor, ki je imel 12 bitne besede in vseboval aritmetično logično enoto, akumulator, programski števec, ukazni register in nekaj kontrolne logike. Zaradi majhnega števila enot je bila njegova fizična razmestitev

zasnovana ročno. Za zagotavljanje pravilne kontrole QCA celic z urinim signalom so avtorji predlagali enako širino za vse urine cone. V posamezni urini coni naj bi se nahajalo manj kot velikostnega reda 10^3 QCA celic. Za omogočanje povratnih povezav so vpeljali t. i. „trapezno geometrijo strukture“ (angl. *trapezoidal floorplan*). Za pomoč pri snovanju in testiranju logičnih struktur QCA so razvili računalniško orodje *Q-BART*. Omogočalo je grafično postavljanje logičnih primitivov in simulacijo delovanja zasnovane logične strukture. Simulacija ni upoštevala urinega signala.

V članku [28] Niemier in Kogge podrobneje obravnavata probleme pri snovanju logičnih struktur QCA, med katerimi so dolžina linij, širina urinih con, število celic v urini coni, povratne povezave in porabljena površina.

Ista avtorja [81] obravnavata tudi cevovodnost (angl. *pipelining*), večnitnost (angl. *multithreading*) in procesiranje v liniji (angl. *processing-in-wire*) pri QCA. Za primer sta uporabila ročno zasnovani mikroprocesor *Simple 12* in predstavila dve splošni geometriji strukture. Zasnovani mikroprocesor sta razširila v cevovodno delujoči *Simple 12 One-Hot*.

Niemier in sodelavci [82] so se ukvarjali tudi s snovanjem FPGA na osnovi QCA. Za osnovni logični gradnik so izbrali majoritetna vrata, ki so realizirala logično operacijo NAND. Predstavili so tudi elemente za povezovanje logičnih gradnikov, sestavljene iz različnih števil urinih con za zagotavljanje pravočasnega prenosa signala.

V doktoratu [20] je Niemier povzel svoje objavljene raziskave. Obravnaval je snovanje mikroprocesorja v QCA in predstavil načrtovalska pravila. Določil je štiri pogoje, ki jim mora zadoščati QCA za uspešno uporabo:

- logična polnost,
- povezovanje na dvodimenzionalni površini in povratne povezave,
- deterministična proizvodnja in
- možnost izdelave fizične razmestitve strukture na osnovi logične sheme.

Po Niemierju je snovanje fizične razmestitve strukture QCA raziskoval Antonelli s sodelavci [84]. Proces snovanja je razdelil v tri faze in sicer razdelitev strukture v urine cone (angl. *partitioning*), razmeščanje logičnih primitivov (angl. *placement*) in njihovo medsebojno povezovanje (angl. *routing*). V članku so se osredotočili le na prvo fazo. Logično strukturo QCA so predstavili z usmerjenim acikličnim grafom, v katerem vozlišča

predstavljajo majoritetna vrata v strukturi, povezave v grafu pa ustrezajo povezavam med majoritetnimi vrati. V procesu so morali zadostiti trem zahtevam:

- zagotoviti istočasen prihod vhodnih signalov do majoritetnih vrat,
- minimizirati število urinih ciklov in s tem zakasnitev signala v strukturi in
- določiti enako višino za vse urine cone, določeno s številom majoritetnih vrat in linij v njej.

Problem so formulirali s celoštevilskim linearnim programom in razvili hevristično metodo za njegovo reševanje.

V članku [21] so avtorji poskušali:

- ugotoviti katere logične primitive je možno podvojiti in s tem zmanjšati število križanj linij,
- preurediti položaje logičnih primitivov za zmanjšanje števila križanj linij,
- minimizirati dolžine povezav za preprečevanje pojava urinega zamika (angl. *clock skew*) in za preprečevanje okvar in napak in
- zmanjšati porabljeno površino za lažjo fizično sestavo strukture.

Avtorji so iskali razmestitev logičnih primitivov z minimalno porabljeno površino, minimalnim številom križanj linij in minimalno skupno dolžino linij. Pri tem so upoštevali naslednje omejitve:

- časovno omejitev (zakasnitev signala v urini coni mora biti manjša od urine periode),
- omejitev položajev priključkov (vhodno / izhodni priključki se morajo nahajati le na zgornji ali spodnji meji logičnega bloka) in
- omejitev smeri signala (signal mora potovati od vhodnih do izhodnih priključkov).

Usmerjeni graf logične sheme so z dodajanjem vmesnih vrat (angl. *feed-through gates*) preslikali v *k-nivojski dvodelni graf*. Tako so dobili vrste, v katerih se nahajajo logični primitivi. Ker se lahko v nekaterih vrstah pojavi veliko število primitivov, so dolge vrste razdelili na vrste enakomerne dolžine z algoritmom zvijanja vrst (angl. *row-folding*). Ker

je problem minimizacije števila križanj linij NP-težak, so za njegovo reševanje uporabili težiščno hevristično metodo za minimizacijo križanj povezav v k-nivojskem dvodelnem grafu.

V delu [31] so minimizirali število križanj linij pri *povezovanju po kanalu* (angl. *channel routing*). Na podlagi priključkov na kanalu so sestavili graf vertikalnih omejitev (angl. *vertical constraint graph*). Iz slednjega so odstranili cikle z rešitvijo problema množice uteženih minimalnih povratnih povezav (angl. *weighted minimum feedback edge set problem*). Problem je NP-poln, zato so za njegovo rešitev predstavili hevristično metodo.

Chung s sodelavci [30] je na podlagi raziskav v delih [21, 31] razdelil algoritem snovanja fizične razmestitve strukture QCA na štiri faze:

- razdelitev strukture na urine cone (angl. *zone partitioning*),
- razmeščanje urinih con (angl. *zone placement*),
- razmeščanje celic in podvajanje (angl. *cell placement and duplication*) in
- povezovanje po kanalu (angl. *channel routing*).

Lim in sodelavci [85] so povzeli objave [21, 31, 30] in poskušali ugotoviti:

- kaj je računsko uporabno in kaj je možno dejansko implementirati,
- katere zasnove struktur bodo uporabne ob napredku tehnologije in
- kako primerjati QCA z današnjimi računalniki na osnovi silicija.

V članku [32] so avtorji postavili domnevo, da bo v molekularnih QCA težko implementirati ravninsko križanje linij. Križanja so skušali odpraviti s podvojevanjem ustreznih logičnih primitivov v strukturi. Problem so formulirali s celoštevilskim linearnim programom in ga reševali s pomočjo hevristične metode.

Teodosio je v članku [23] in v svojem magisteriju [88] predstavil računalniško orodje *QCA-LG*. Orodje na osnovi opisa vezja v jeziku VHDL avtomatično izdela fizično razmestitev strukture QCA. To je mogoče nato še ročno optimizirati in simulirati z orodjem QCADesigner. Vsa križanja linij v fizični razmestitvi strukture se odstranijo s podvojevanjem ustreznih logičnih primitivov.

Vankamamidi in sodelavci [89] so skušali razdeliti strukturo QCA na mrežo z urinimi conami, tako da bi lahko signal potekal v vodoravni in navpični smeri. To bi lahko zmanjšalo dolžino najdaljše linije v posamezni urini coni.

Avtorji so v delu [24] obravnavali snovanje fizične razmestitve strukture QCA s poudarkom na upoštevanju dejavnikov, ki omogočajo snovanje izvedljivih struktur. Pri tem so kritizirali nekatere dotedanje raziskave:

- v delu [84] ne upoštevajo geometrije fizične razmestitve strukture in termodinamičnih dejavnikov;
- v delu [85] ne izračunajo dejanskih dolžin linij, kar lahko privede do strukture, ki ne deluje logično pravilno;
- v delu [23] se struktura s podvojevanjem logičnih primitivov preslika v trojiško drevo, pri čemer se število izhodnih linij logičnih primitivov potroji. To privede do eksponentnega povečanja strukture QCA.

V članku [33] so avtorji predlagali ravninsko križanje linij z uporabo enega samega tipa celic. V ta namen so vpeljali tri tipe 8 faznih urinih signalov. Križanje poteka signalov so izvedli s časovnim multipleksiranjem na podlagi nove urine sheme.

3 Izhodišča za razmeščanje in povezovanje logičnih primitivov v QCA

3.1 Osnove fizičnega snovanja logične strukture

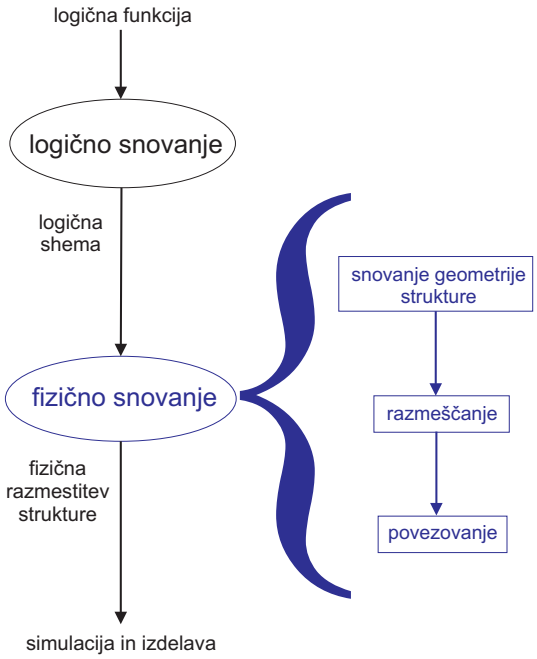
Mnogo problemov lahko opišemo z logičnimi funkcijami. Kompleksne logične funkcije je v praksi izjemno težko izračunati brez uporabe tehnologije. Zato se funkcije realizirajo z logičnimi strukturami, ki se lahko fizično izdelajo. Pred izdelavo je potrebno logično strukturo zasnovati. Pri tem se morajo upoštevati dane zahteve in omejitve tehnologije, hkrati pa se optimizirajo določene lastnosti logične strukture.

Postopek *snovanja logične strukture* se v grobem deli na dva dela [8]:

- *logično snovanje* (angl. *logic design*) in
- *fizično snovanje* (angl. *physical design*).

Rezultat logičnega snovanja je logični opis strukture, ki je podan z logično shemo in z logičnimi izrazi za izračun izhodov. Funkcija se med postopkom optimizira glede na več kriterijev. Eden od kriterijev, ki se v logičnem snovanju minimizira, je na primer število operacij iz izbranega nabora. V tem delu logičnega snovanja ne bomo obravnavali in bomo privzeli, da je logični opis strukture že podan.

Osredotočili se bomo na fizično snovanje, katerega namen je določiti razmestitev gradnikov logične strukture in jih med seboj povezati. Rezultat postopka je *fizična razmestitev strukture*, ki natančno določa položaje logičnih primitivov in lego povezav med njimi. Na podlagi fizične razmestitve je možno simulirati delovanje strukture in jo končno fizično izdelati. Predstavili bomo tri glavne stopnje fizičnega snovanja in sicer *snovanje geometrije strukture* (angl. *floorplanning*), *razmeščanje* (angl. *placement*) logičnih primitivov in njihovo medsebojno *povezovanje* (angl. *routing*). Proces snovanja logične strukture je shematično prikazan na sliki 3.1.



Slika 3.1 Proces snovanja logične strukture. Na desni strani slike so prikazane tri glavne stopnje fizičnega snovanja.

3.1.1 Snovanje geometrije strukture

V postopku logičnega snovanja se določijo logični operatorji, ki jih bodo realizirali logični primitivi. Slednji se *porazdelijo* (angl. *partition*) v skupine glede na izbran kriterij [11]. Izbrali bomo takšen kriterij porazdelitve primitivov v skupine, da ne bo obstajal signal, ki bi potoval od izhoda enega do vhoda drugega primitiva v isti skupini. To pomeni, da logični primitivi v isti skupini med seboj niso povezani. Označimo množico primitivov s P=$\{p_1, p_2, ..., p_{N_1}\}$ in množico skupin s S=$\{S_1, S_2, ..., S_{N_2}\}$. Podanih imamo

N_1 logičnih primitivov, ki jih bomo porazdelili v N_2 skupin. Vsaka skupina S_i vsebuje neko podmnožico vseh primitivov P_i:

$$S_i = \{p|p \in P_i, P_i \subseteq \mathrm{P}\}. \tag{3.1}$$

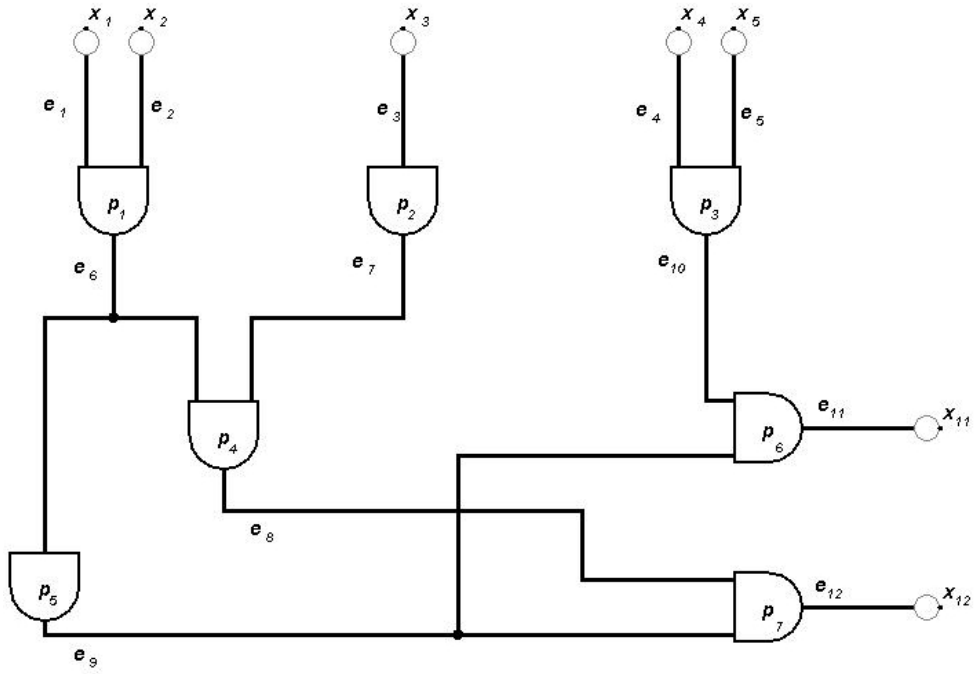
Množice P_i so med seboj disjunktne za vse indekse i. Če med primitivoma p_i in p_j obstaja povezava in $p_i \in S_k$, potem mora veljati $p_j \notin S_k$.

Površina, na katero bo nameščena logična struktura, se razdeli na pravokotna področja [11]. Vsaki skupini ustreza eno področje, znotraj katerega bodo nameščeni vsi primitivi v skupini. Ker imajo primitivi določene oblike, lahko določimo tudi obliko namestitvenega področja, tako da bo lahko vsebovalo vse pripadajoče primitive. Vzemimo množico področij B=$\{b_1, b_2, ..., b_{N_2}\}$. Vsako področje $b_i \in$ B ima določeni širino w_i in višino h_i. Slednji morata biti zadosti veliki, da se lahko v področje namestijo vsi pripadajoči logični primitivi v skupini S_i brez prekrivanja. Skupini S_i ustreza področje b_i.

Točko, v kateri se povezava priključi na vhod logičnega primitiva, imenujemo *vhodni priključek*. Podobno je točka, v kateri se povezava priključi na izhod logičnega primitiva, imenovana *izhodni priključek*. Na podlagi logične sheme lahko sestavimo seznam povezav E=$\{e_1, e_2, ..., e_{N_3}\}$ med vsemi priključki v strukturi. Množico E sestavljata disjunktni podmnožici E_1 in E_2, tako da velja $\mathrm{E}_1 \cup \mathrm{E}_2 = \mathrm{E}$ in $\mathrm{E}_1 \cap \mathrm{E}_2 = \emptyset$. Vsaka povezava v E_1 povezuje en izhodni priključek na enem primitivu s poljubnim številom vhodnih priključkov na enem ali več preostalih primitivih. Povezava v E_2 povezuje ali en vhodni priključek strukture in vhodne priključke na logičnih primitivih ali en izhodni priključek na primitivu z enim izhodnim priključkom strukture. Razvejitev signala omogočajo razvejitvene povezave. Ker priključki v istem področju med seboj niso povezani, vsaka povezava $e_i \in \mathrm{E}_1$ poteka od izhodnega priključka v enem področju do enega ali več vhodnih priključkov v preostalih področjih. Tako si lahko množico E_1 predstavljamo tudi kot seznam povezav med področji.

Za primer vzemimo logično shemo na sliki 3.2. Množico logičnih primitivov P v tem primeru sestavljajo p_1, p_2, p_3, p_4, p_5, p_6 in p_7. Vhodni priključki strukture so x_1, x_2, x_3, x_4 in x_5, izhodna priključka pa sta y_1 in y_2. Množico povezav E=$\{e_1, e_2, ..., e_{12}\}$ sestavljata podmnožici E_1=$\{e_6, e_7, e_8, e_9, e_{10}\}$ in E_2=$\{e_1, e_2, e_3, e_4, e_5, e_{11}, e_{12}\}$.

Vhodna podatka za snovanje geometrije strukture sta množica pravokotnih področij B in množica povezav E. Cilj postopka je razporediti področja na površino brez prekrivanja.



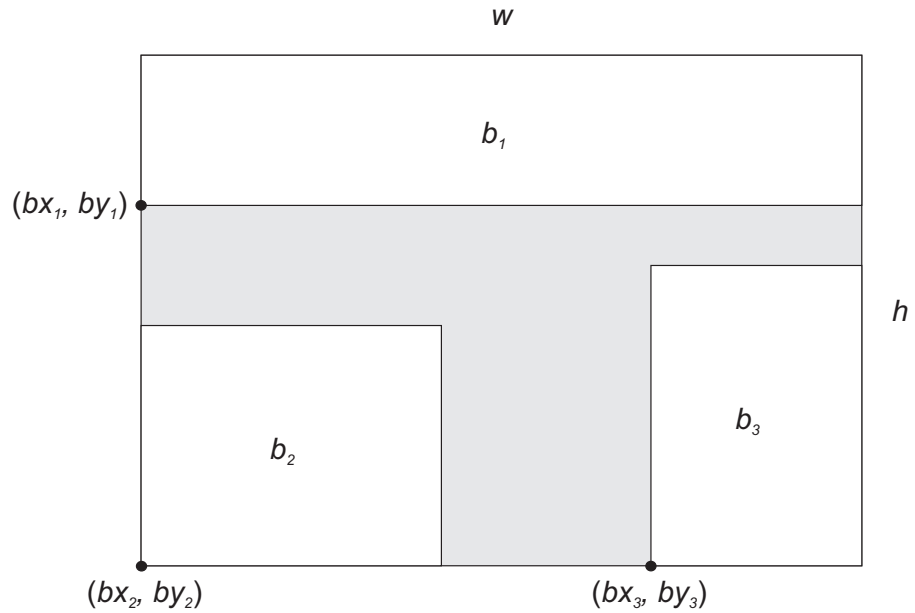
Slika 3.2 Primer logične sheme.

Pri tem je potrebno upoštevati podane omejitve, kot sta na primer maksimalni dimenziji celotne porabljene površine. Rezultat postopka so natančno določeni položaji področij na površini. Če koordinata (bx_i, by_i) označuje koordinato spodnjega levega oglišča področja b_i, je rezultat množica koordinat

$$\mathrm{B}_{koordinate} = \{(bx_i, by_i) | 1 \leq i \leq N_2\}. \tag{3.2}$$

Med področji se lahko pusti nekaj *proste površine* (angl. *channel*), na katero se namestijo povezave [11]. Na prosti površini se izvedejo tudi križanja povezav. Primer razdelitve pravokotne površine s stranicama w in h na tri področja b_1, b_2 in b_3 prikazuje slika 3.3. Med področji se nahaja prosta površina za namestitev povezav.

Med procesom določanja položajev področij se optimizira več kriterijev. Med njimi sta velikost celotne porabljene površine in ocena skupne dolžine povezav. Navedeni količini se skušata minimizirati. Pomembna je tudi čimbolj enakomerna porazdelitev povezav na površini, saj območja z veliko gostoto povezav otežujejo njihovo nameščanje. V stopnji snovanja geometrije strukture se izračunajo ocene optimalnih vrednosti kriterijev. Vrednosti se lahko med procesom fizičnega snovanja še spreminjajo.



Slika 3.3 Področja b_1, b_2 in b_3, nameščena na pravokotno površino s stranicama w in h. S sivo obarvana prosta površina je namenjena za namestitev povezav.

3.1.2 Razmeščanje logičnih primitivov

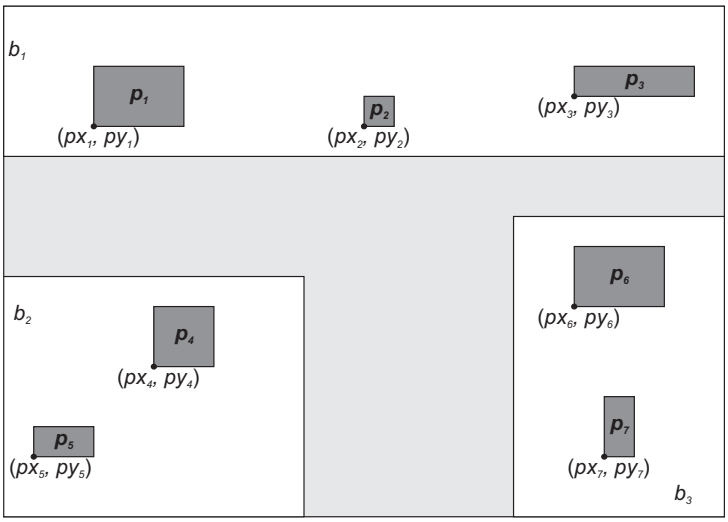
Množica B pravokotnih področij z določenimi dimenzijami in množica S skupin z logičnimi primitivi predstavljata osnovo za stopnjo razmeščanja. Podani sta tudi množica primitivov P in množica E povezav med priključki. Na področje $b_i \in$ B morajo biti nameščeni primitivi iz množice $S_i \in$ S. Cilj postopka razmeščanja je namestitev logičnih primitivov v ustrezna področja. Pri tem se primitivi med seboj ne smejo prekrivati in ne smejo segati izven meja pripadajočega področja. Rezultat razmeščanja je množica koordinat

$$\mathrm{P}_{koordinate} = \{(px_i, py_i) | 1 \leq i \leq N_1\}, \tag{3.3}$$

kjer (px_i, py_i) označuje koordinato spodnjega levega oglišča logičnega primitiva p_i.

Slika 3.4 prikazuje razmestitev sedmih logičnih primitivov p_1, p_2, p_3, p_4, p_5, p_6 in p_7 v treh področjih b_1, b_2 in b_3. Položaji področij ustrezajo geometriji, predstavljeni na sliki 3.3. Primitivi so razdeljeni v skupine $S_1 = \{p_1, p_2, p_3\}$, $S_2 = \{p_4, p_5\}$ in $S_3 = \{p_6, p_7\}$, tako da skupini S_i ustreza področje b_i.

Podobno kot pri snovanju geometrije strukture se tudi pri razmeščanju optimizirata skupna dolžina povezav in porazdelitev gostote povezav na površini. Poleg tega se mini-



Slika 3.4 Razmestitev sedmih logičnih primitivov p_1, p_2, p_3, p_4, p_5, p_6 in p_7 v treh področjih b_1, b_2 in b_3.

mizirata število križanj povezav in dolžine najdaljših povezav. Optimizirane vrednosti so zaenkrat le ocene končnih vrednosti, ki se dokončno določijo šele v stopnji povezovanja.

Optimalnost razmestitve primitivov je odvisna od zasnovane geometrije strukture. Če rešitev ni zadovoljiva in je ni mogoče izboljšati v fazi razmeščanja, se lahko ponovno izvede postopek snovanja geometrije logične strukture. Na ta način se določi primernejšo razporeditev področij, na podlagi katere se znova poišče optimalno razmestitev logičnih primitivov.

3.1.3 Povezovanje logičnih primitivov

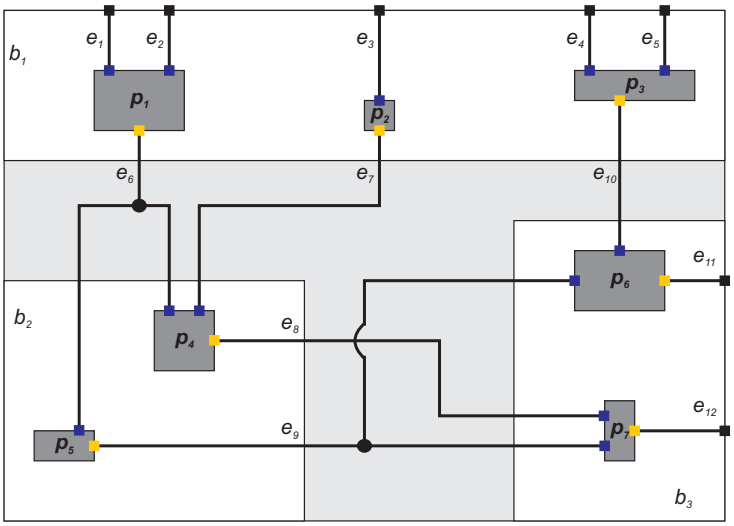
Prvi vhodni podatek za fazo povezovanja je optimizirana razmestitev logičnih primitivov iz množice P. Vsak primitiv $p_i \in \mathrm{P}$ ima določen položaj na površini s koordinato $(px_i, py_i) \in \mathrm{P}_{koordinate}$. Vhodni podatki so še množica področij B, množica koordinat področij $\mathrm{B}_{koordinate}$ in množica povezav $\mathrm{E} = \mathrm{E}_1 \cup \mathrm{E}_2$. Vsaka povezava $e_i \in \mathrm{E}_1$ povezuje en izhodni priključek na enem primitivu z enim ali več vhodnih priključkov na ostalih primitivih. Navedeni priključki pripadajo povezavi e_i. Množica E_2 vsebuje povezave med vhodno izhodnimi priključki logične strukture in priključki na logičnih primitivih.

Namen povezovanja logičnih primitivov je namestitev vseh povezav v množici E na po-

vršino. Vsaka povezava mora povezati vse pripadajoče priključke. Pri tem se upoštevajo dane omejitve, kot je na primer minimalna potrebna razdalja med dvema vzporednima povezavama. Rezultat postopka je natančno določena lega vsake povezave.

Del povezave, ki se nahaja znotraj področja, poteka do njegovega roba v najkrajši liniji. Povezave znotraj področja se med seboj ne križajo, niti ne potekajo čez območja, ki jih zasedajo logični primitivi. Del povezave v področju povezuje priključek na logičnem primitivu s točko na robu področja. Na zunanjih robovih se nahajajo vhodni in izhodni priključki strukture. Povezave med priključki strukture in ustreznimi priključki na logičnih primitivih potekajo znotraj področja. Ko so nameščene povezave znotraj področij, preostane še povezovanje ustreznih točk na robovih področij, ki se izvede z nameščanjem povezav na prosto površino. Na slednji se implementirajo križanja povezav [11].

Slika 3.5 prikazuje nameščene povezave med logičnimi primitivi p_1, p_2, p_3, p_4, p_5, p_6 in p_7. Razmestitev primitivov v področjih b_1, b_2 in b_3 je enaka kot na sliki 3.4.



Slika 3.5 Povezana razmestitev logičnih primitivov s slike 3.4. Področja so označena z b_i, logični primitiv s p_i in povezave z e_i. Vhodni priključki strukture se nahajajo na zgornjem robu področja b_1, izhodni pa na desnem robu b_3. Oboji so označeni s črnimi kvadratki. Vhodne priključke na logičnih primitivih označujejo modri kvadratki in izhodne priključke rumeni. Razvejitvi povezav e_6 in e_9 označuje črn krog. Prikazano je tudi križanje povezav e_8 in e_9.

Prikazanih je dvanajst povezav iz množice E=$\{e_1, e_2, ..., e_{12}\}$, ki jo sestavljata podmno-

žici $E_1=\{e_6, e_7, e_8, e_9, e_{10}\}$ in $E_2=\{e_1, e_2, e_3, e_4, e_5, e_{11}, e_{12}\}$. Označeni so tudi vhodni in izhodni priključki strukture ter vsi priključki na logičnih primitivih.

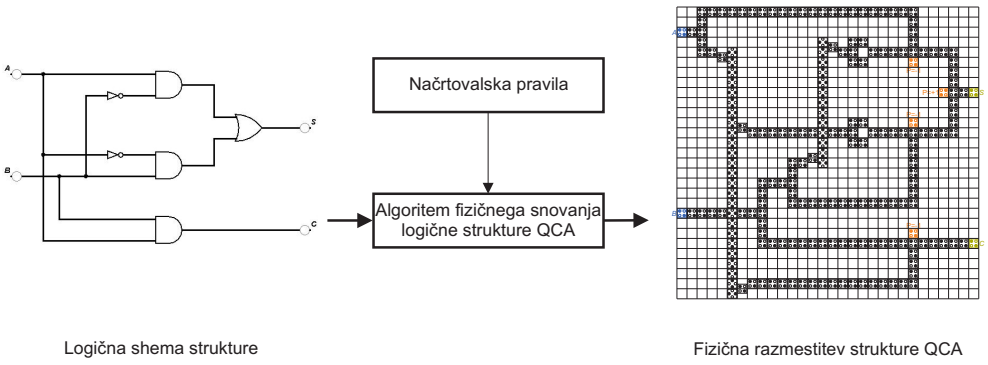
Rezultat povezovanja so tudi dokončno določene optimizirane vrednosti kriterijev, kot so skupna dolžina povezav, število križanj povezav, porazdelitev gostote povezav na površini, itd. Kvaliteta rezultata faze povezovanja je odvisna od vhodnih podatkov. Lahko se zgodi, da rešitev ni dovolj kvalitetna, ali pa celo sploh ni možno namestiti vseh povezav v skladu z zahtevami. Takrat se ponovno izvede postopek razmeščanja logičnih primitivov ali pa tudi snovanje geometrije strukture. Išče se taka rešitev, da bo pri ponovnem povezovanju možno najti boljši rezultat.

3.2 Fizično snovanje logične strukture na osnovi QCA

Cilj pričujoče doktorske disertacije je vzpostavitev avtomatiziranega postopka fizičnega snovanja QCA logične strukture. Iščemo metodo, ki bo realizirala želeni cilj z izvedbo procesa fizičnega snovanja logične strukture. V obstoječih raziskavah [90, 37, 75, 91, 5, 92, 6, 93] so bile ročno zasnovane nekatere enostavne QCA logične strukture. Tako snovanje je potekalo z ad hoc postopkom, ki ga ni mogoče posplošiti za uporabo pri snovanju poljubne strukture. Za vpeljavo avtomatičnega snovanja bo potrebno metodo formalizirati in jo zapisati z algoritmom. Vhod algoritma bo logična shema strukture, izhod (rezultat) pa fizična razmestitev strukture QCA. Slednja je definirana z natanko določenimi legami QCA logičnih primitivov v dvodimenzionalnem prostoru. V fizični razmestitvi strukture QCA je tako za vsako QCA celico znan njen položaj v pravokotni površinski mreži. Avtomatizirano snovanje bo potekalo z upoštevanjem načrtovalskih pravil. Z njimi so določene zahteve, ki jim mora zadoščati fizična razmestitev strukture QCA, tako da bo simulacija njenega delovanja dala pričakovane rezultate. Vhod in izhod algoritma fizičnega snovanja logične strukture QCA sta shematično prikazana na sliki 3.6.

Isto logično funkcijo lahko realizira več različnih fizičnih razmestitev strukture QCA. Razlike med slednjimi se kažejo v njihovih lastnostih. Nekatere lastnosti so bolj zaželene kot druge, zato lahko na njihovi podlagi postavimo kriterije za ovrednotenje posamezne fizične razmestitve QCA. Vzpostavljena metoda bo izvajala večkriterijsko optimizacijo, pri čemer bodo minimizirane vrednosti naslednjih kriterijev:

- *število potrebnih urinih faz za izračun rezultata,*



Slika 3.6 Shematični prikaz vhoda in izhoda algoritma fizičnega snovanja logične strukture QCA.

- *vsota dolžin vseh linij*,
- *število kotnih linij*,
- *porabljena površina* in
- *število križanj linij*.

Kriteriji so med seboj soodvisni, kar pomeni, da lahko zmanjšanje vrednosti enega izmed njih povzroči pozitivno ali negativno spremembo vrednosti ostalih kriterijev.

Vzpostavljena metoda bo morala ob optimizaciji navedenih kriterijev z upoštevanjem načrtovalskih pravil doseči naslednje cilje:

- *razdeliti logične primitive v urine cone*,
- *določiti število potrebnih urinih faz*,
- *razmestiti logične primitive v posamezne urine cone* in
- *medsebojno povezati logične primitive v strukturi*.

3.3 Zasnova geometrije strukture

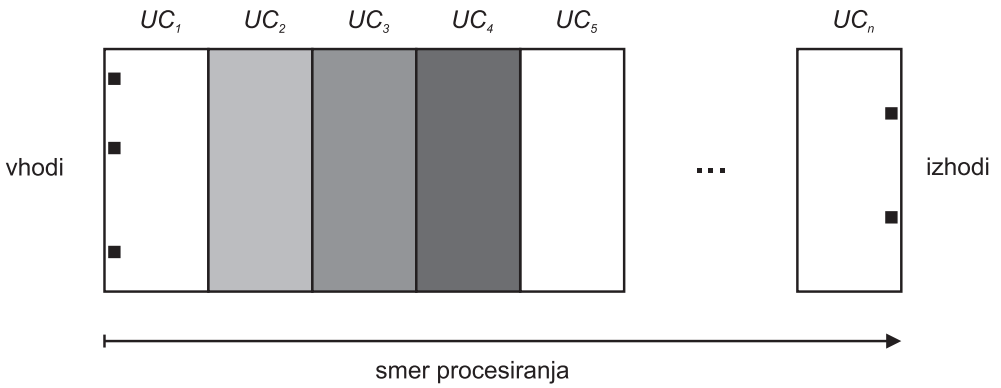
Kompleksno CMOS vezje za generiranje urinega signala lahko zasenči prednosti, ki jih ponuja QCA tehnologija pred obstoječimi sistemi. Možna hitrost preklopa stanja QCA celice je nekaj velikostnih redov večja od hitrosti preklopa v CMOS vezju [20]. Zato je potrebno zasnovati čimbolj enostavno CMOS vezje, ki bo omogočalo čim večjo frekvenco urinega signala. To ni zaželeno le zaradi povečanja hitrosti delovanja QCA, pač pa je tudi

nujno za zagotovitev prenosa pravilnega logičnega stanja QCA celice, preden se njegova vrednost izgubi [20].

Dodatno omejitev predstavlja razlika med velikostmi elementov v CMOS in QCA tehnologiji. Dimenzije CMOS elektrode so večje od velikosti QCA celice [94]. Ker je širina urine cone določena z velikostjo elektrode pod njo, mora biti urina cona nekajkrat večja od velikosti ene QCA celice. Tako ima linija znotraj ene urine cone določeno minimalno dolžino, ki ustreza širini cone. Hkrati pa linija ne sme vsebovati več kot N_{max} QCA celic, kar navzgor omejuje širino posamezne urine cone.

Minimalni čas trajanja urinega cikla določa T_{cikel}. Ta čas je sorazmeren s številom celic v liniji, ki jih kontrolira isti urin signal. Čas preklopa je tako odvisen od števila celic v najdaljši liniji znotraj urine cone. Za preprečevanje pojava urinega zamika želimo enak čas preklopa v vseh conah. Zato smo določili enotno velikost vseh urinih con in s tem poenotili tudi dolžino najdaljše linije znotraj posamezne cone.

Na podlagi navedenih lastnosti QCA smo določili pravilno, omejeno in enotno velikost urine cone. Vsaka cona ima pravokotno obliko s fiksno širino. Celotna fizična razmestitev strukture QCA je razdeljena na n urinih con, ki so razporejene od vhodov na levi do izhodov na desni strani. Procesiranje poteka od vhodov proti izhodom, kot je prikazano na sliki 3.7.



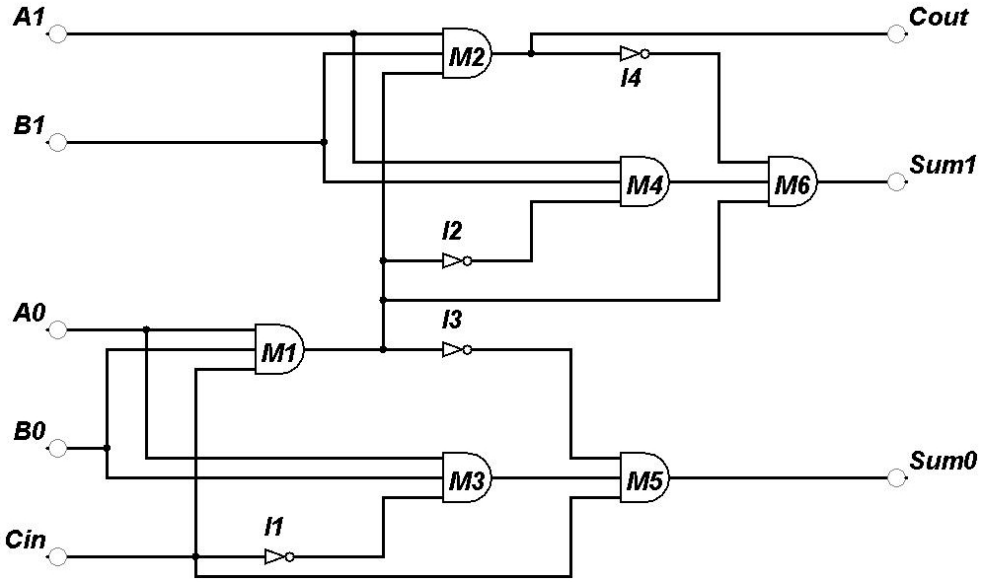
Slika 3.7 Razporeditev urinih con UC_1, UC_2, ..., UC_n in smer poteka procesiranja v fizični razmestitvi strukture QCA.

3.4 Opisi ciljev

3.4.1 Razdelitev logičnih primitivov v urine cone

Vhodni podatek metode za snovanje fizične razmestitve strukture je logična shema strukture kombinatoričnega vezja, na podlagi katere lahko sestavimo *usmerjeni graf logične sheme*. Kombinatorično vezje nima spomina in zato tudi ne vsebuje povratnih povezav. Izhod je odvisen le od trenutnega vhoda in ne tudi od prejšnjih vhodov.

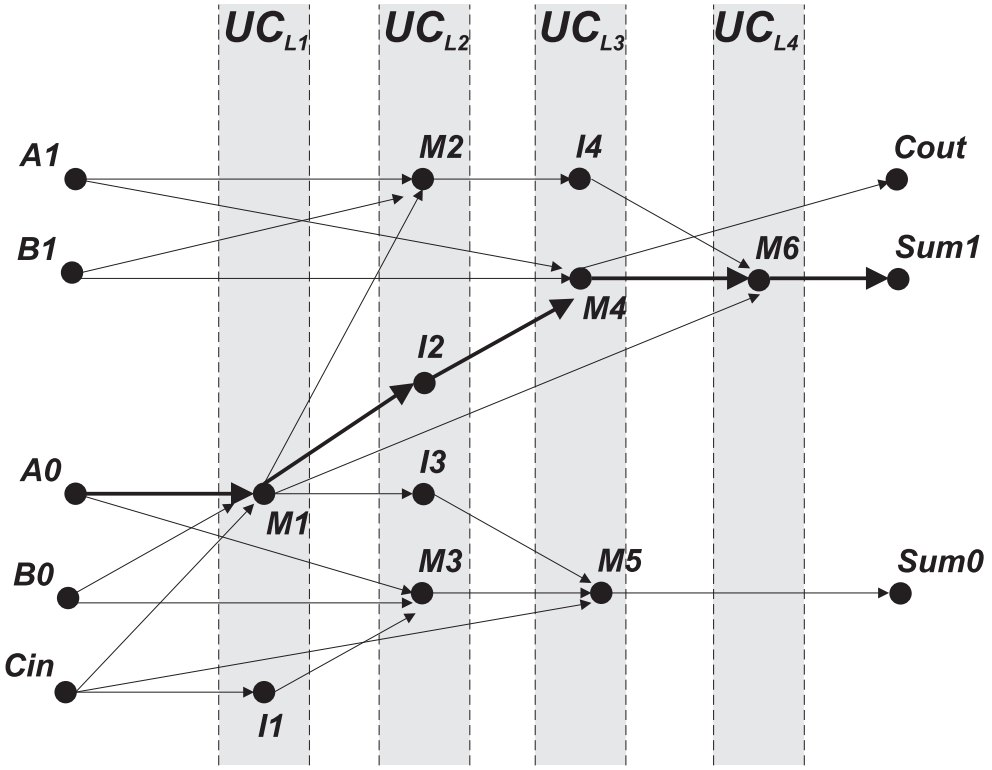
Kot zgled vzemimo logično shemo strukture dvobitnega polnega seštevalnika, ki je prikazana na sliki 3.8, pripadajoči usmerjeni graf pa na sliki 3.9. Vozlišča v grafu



Slika 3.8 Logična shema dvobitnega polnega seštevalnika, sestavljenega iz šestih majoritetnih vrat ($M1$, $M2$, $M3$, $M4$, $M5$ in $M6$) in štirih negatorjev ($I1$, $I2$, $I3$ in $I4$).

ustrezajo logičnim primitivom v shemi, usmerjene povezave med vozlišči pa so določene s povezavami med primitivi v shemi. Usmerjeni graf logične sheme kombinatoričnega vezja ne vsebuje ciklov, zato ga lahko topološko uredimo [95].

V vsakem topološko urejenem usmerjenem grafu obstaja ena ali več *kritičnih poti*, ki imajo vse enako dolžino. Kritična pot je najdaljša pot v usmerjenem grafu [95]. Dolžina kritične poti p določa število urinih con, ki vsebujejo logične primitive. Take cone označimo z UC_{L1}, UC_{L2}, ..., UC_{Lp}. Na sliki 3.9 so štiri cone z logičnimi primitivi, označene z UC_{L1}, UC_{L2}, UC_{L3} in UC_{L4}. Vsako vozlišče na kritični poti lahko pripada le



Slika 3.9 Usmerjeni graf logične sheme s slike 3.8. Kritična pot od vhodnega vozlišča $A0$ do izhodnega $Sum1$ je predstavljena z odebeljenimi puščicami. Vozlišča, ki predstavljajo logične primitive, so razdeljena v štiri urine cone UC_{L1}, UC_{L2}, UC_{L3} in UC_{L4}.

eni urini coni, ki je določena s položajem vozlišča na poti. Ostala vozlišča so "plavajoča" in jih lahko umestimo v poljubno cono, pri čemer mora še vedno veljati topološka urejenost med vozlišči. V primeru na sliki 3.9 lahko na primer vsako od vozlišč $I3$, $M3$ in $M5$ prestavimo za eno cono naprej, tako da vstavimo vozlišči $I3$ in $M3$ v cono UC_{L3}, vozlišče $M5$ pa v cono UC_{L4}. Če pa $M5$ ostane v UC_{L3}, se vozlišči $I3$ in $M3$ ne smeta premakniti iz cone UC_{L2}.

3.4.2 Določanje števila potrebnih urinih faz

Med dvema zaporednima conama z logičnimi primitivi UC_{Li} in UC_{Li+1} lahko vstavimo več vmesnih urinih con, v katerih so implementirane povezave med primitivi. Število vmesnih urinih con je odvisno od lastnosti povezave med celico na desnem robu UC_{Li} in celico na levem robu cone UC_{Li+1}. Če je linija povsem ravna, vmesna cona ni potrebna, ker se logična vrednost prenese neposredno z izhodne celice logičnega primitiva v coni

UC_{Li} na vhodno celico primitiva v coni UC_{Li+1}. Kadar pa je za povezavo potrebna kotna linija, mora biti ta nameščena v vmesni coni. Podobno je v primeru razvejitvene linije, ki prav tako zahteva vstavitev vmesne cone. V slednjih je implementirano tudi križanje linij. Če razdalje med križajočimi se linijami ne zadoščajo načrtovalskim pravilom, to lahko zahteva vstavitev večjega števila vmesnih urinih con.

Pri r dodanih urinih conah vsebuje struktura $p + r$ urinih con. S tem je določena tudi zakasnitev signala v strukturi, ki znaša $n = p + r$ urinih faz. Fizična razmestitev strukture je razdeljena na p urinih con z logičnimi primitivi UC_{Li} $(1 \leq i \leq p)$ in r vmesnih con UC_{Vj} $(0 \leq j \leq r)$.

3.4.3 Razmeščanje primitivov v posamezne urine cone

Ko so določene urine cone, v katerih se logični primitiv lahko nahaja, se mora določiti še njegov položaj znotraj ene izmed možnih con. Za razmeščanje bomo uporabili tehniko simuliranega ohlajanja. Pri tem se minimizirajo ocena skupne dolžine linij, ocena števila kotnih linij in ocena števila križanj linij. Upoštevati se morajo tudi načrtovalska pravila. Eno izmed teh na primer določa minimalno zadostno razdaljo med logičnimi primitivi.

Rezultat te stopnje so določeni položaji logičnih primitivov znotraj urinih con in s tem znotraj strukture, pri čemer so vrednosti optimizacijskih kriterijev čim manjše.

3.4.4 Medsebojno povezovanje logičnih primitivov v strukturi

Za prenos signalov je potrebno ustrezne logične primitive med seboj povezati z linijami. Za to se uporabljata algoritma iskanja po labirintu in iskanja po liniji. Prvi temelji na iskanju v širino, drugi pa na iskanju v globino. Za povezovanje bomo uporabili kombinacijo obeh algoritmov. Najprej se globalno poišče prostor za namestitev povezave z iskanjem po labirintu, nato pa se podrobno določi lega povezave z iskanjem po liniji. Kriteriji optimizacije pri povezovanju so vsota dolžine vseh linij, število kotnih linij, število križanj linij, poraba površine in število potrebnih urinih faz.

Rezultat povezovanja so določene lege linij med logičnimi primitivi. Šele v tej stopnji se ugotovijo dejanske dolžine linij, število kotnih linij, število križanj, porabljena površina in število potrebnih urinih faz. Določi se končna višina urinih con.

4 Načrtovalska pravila

4.1 Uvod

Pred razvojem računalniškega orodja za avtomatizirano fizično snovanje logične strukture je najprej potrebno formalizirati metodologijo snovanja. Nato se lahko na podlagi formalne specifikacije implementirajo algoritmi za avtomatizacijo procesa snovanja. Pravilno delujoče strukture je možno zasnovati le z uporabo *načrtovalskih pravil* (angl. *design rules*), ki jih mora zato upoštevati tudi računalniško orodje. Mere, določene z načrtovalskimi pravili, so odvisne od tehnologije realizacije strukture. Podane so s številom enot določene dolžine, tako da se lahko zasnove skalirajo glede na izbrano tehnologijo.

Enotno metodologijo snovanja CMOS vezij in njim primerna načrtovalska pravila sta vpeljala avtorja Mead in Conway [7]. Z načrtovalskimi pravili v kvantnih celičnih avtomatih se je ukvarjal Niemier [20]. Vpeljava načrtovalskih pravil in računalniške podpore služi tudi povezovanju različnih znanstvenih disciplin, potrebnih za izgradnjo računalniških struktur. Elektrotehniki in kemiki nimajo potrebnega znanja za snovanje računalniških sistemov. Nasprotno sistemski arhitekti to znanje imajo, ne poznajo pa zadosti podrobnosti o fizični izdelavi naprav. S pomočjo računalniškega orodja in omejitev,

podanih z načrtovalskimi pravili, se izboljša komunikacija med znanstveniki z različnih področij, kar omogoča snovanje in izdelavo vse boljših računalniških sistemov.

Za računalniško podprto snovanje fizične razmestitve QCA smo postavili točno določena načrtovalska pravila, ki jih mora računalniško orodje upoštevati v procesu snovanja [96]. Pravila postavljajo omejitve pri razmestitvi logičnih primitivov v fizični realizaciji strukture. Z njimi so določene potrebne razdalje med posameznimi logičnimi primitivi. Določajo tudi razdelitev linij v urine cone.

4.2 Zahteve pri načrtovalskih pravilih

Načrtovalska pravila definirajo lastnosti, ki jih mora imeti fizična razmestitev strukture, zato da je njeno delovanje pravilno. Pri tem je potrebno upoštevati posebnosti tehnologije QCA. Poleg tega zahteve določa tudi predstavljena zasnova geometrije vezja. Zahteve v načrtovalskih pravilih so predstavljene v naslednjih alinejah:

- Za pravilen prenos signala po QCA celicah mora preklop slednjih kontrolirati ciklično urin signal v pravilnem zaporedju urinih faz. Ko se signal pojavi na vhodnem robu urine cone, mora QCA celice v njej kontrolirati urin signal v fazi preklopa. Zato morajo biti v pravilnem zaporedju razvrščene tudi urine cone, skozi katere potekajo signali. Pravilen vrstni red urinih faz v zaporednih urinih conah je faza preklopa, faza sproščenosti, faza sproščanja in faza zadrževanja.

- Urine cone imajo pravokotno obliko. Širina in višina urinih con se merita s številom področij v mreži, na katerih so lahko nameščene QCA celice. Širino urine cone UC označimo z $W(UC)$, njeno višino pa s $H(UC)$. Določene so robne vrednosti dimenzij, ki jih lahko ima katerakoli urina cona. Če je minimalna širina označena z W_{min} in maksimalna z W_{max}, velja

$$W_{min} \leq W(UC) \leq W_{max}. \tag{4.1}$$

  Podobno za minimalno višino urine cone H_{min} in maksimalno višino H_{max} velja

$$H_{min} \leq H(UC) \leq H_{max}. \tag{4.2}$$

- Vsi signali na linijah, ki se nahajajo v urini coni, potekajo v isti smeri. Vhodni in izhodni priključek na liniji ne smeta biti nameščena na istem robu urine cone.

Signal lahko preide iz vodoravne na navpično in nato spet na vodoravno linijo, zato ni nujno, da sta vhodni in izhodni priključek povezana z ravno linijo. Na ta način lahko signal potuje na primer od leve strani proti desni in ob prehodu na navpično linijo spremeni smer potovanja navzgor. Ko spet preide na vodoravno linijo, lahko signal potuje ponovno proti desni, nikakor pa ne proti levi strani.

- Rezultat osnovnih logičnih operacij naj se izračuna v eni urini fazi. Tako mora biti logični primitiv, ki izvaja logično operacijo, nameščen v eni urini coni. Rezultat logične operacije na izhodnem robu urine cone je izračunan v isti urini fazi, v kateri se na vhodnem robu urine cone pojavijo vhodni signali.

  V nadaljevanju poglavja bomo opisali načrtovalska pravila za logične primitive negator, majoritetna vrata in majoritetna vrata povezana z negatorjem. Vsakega od naštetih gradnikov lahko namestimo v eno urino cono in pri tem zagotovimo želeno delovanje. S tem je mogoče rezultate logičnih operacij NOT, AND, OR, NAND, NOR in majoritetne funkcije izračunati v eni urini fazi.

- Logični primitivi morajo ohraniti pravilno delovanje, kadar vse linije z vhodnimi signali pridejo z iste strani. Pri enostavni razmestitvi urinih con si te sledijo od leve proti desni. Tako se vhodni signali vsake urine cone nahajajo na njeni levi strani, zato tudi vse vhodne linije logičnih primitivov v urini coni potekajo od leve strani.

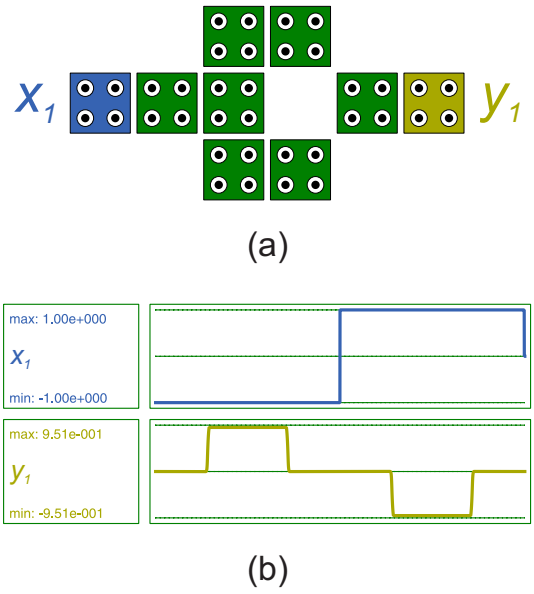
  V nadaljevanju poglavja bomo opisali načrtovalska pravila za logične primitive negator, majoritetna vrata in majoritetna vrata povezana z negatorjem. Z upoštevanjem teh načrtovalskih pravil našteti logični primitivi delujejo pravilno, kadar vse linije z vhodnimi signali pridejo z iste strani.

- Signal lahko potuje od enega vhodnega priključka do več izhodnih. V ta namen se uporablja razvejitev linije (angl. *fanout*). Opisali bomo načrtovalska pravila za izvedbo razvejitvene linije v QCA. Pokazali bomo, da se razvejitev izvede v eni urini fazi.

- Za realizacijo kompleksnejših QCA struktur je potreben mehanizem križanja linij. Med različnimi možnostmi implementacije križanja bomo obravnavali večnivojsko križanje linij. Opisali bomo načrtovalska pravila, ki bodo določala ustrezne razdalje med križajočimi se linijami, tako da bo struktura QCA delovala pravilno.

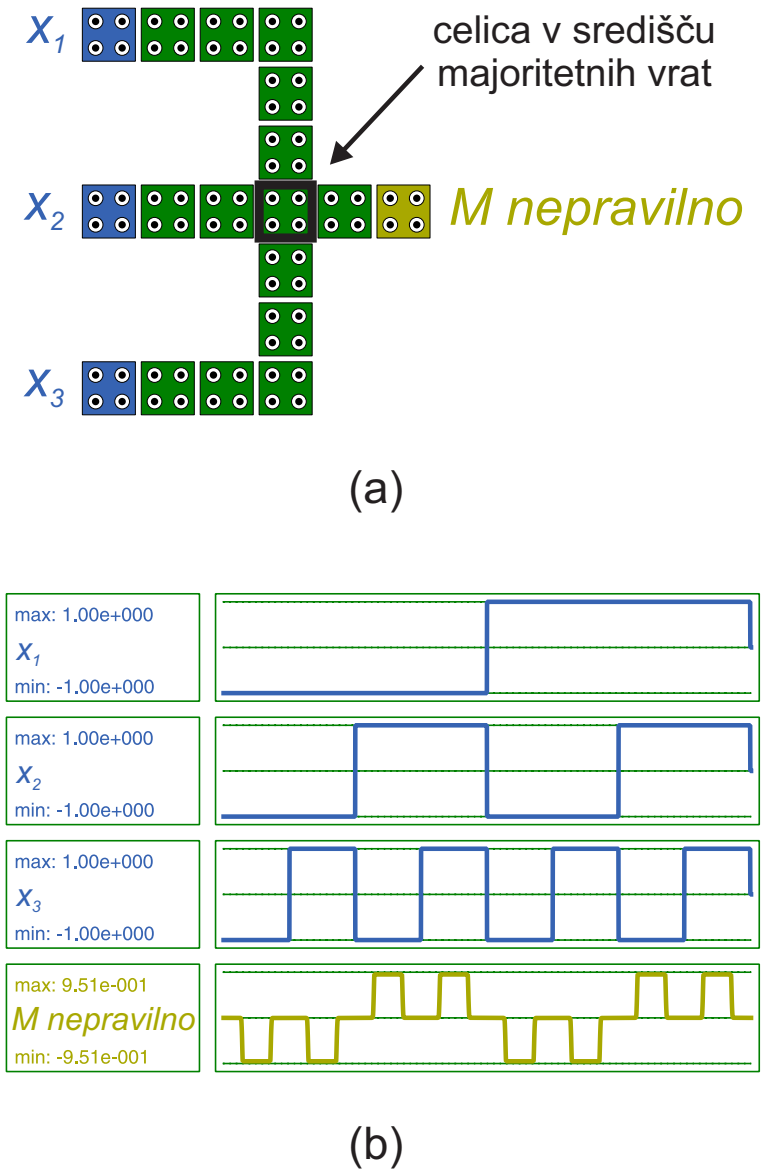
4.3 Načrtovalska pravila za realizacijo logičnih primitivov v eni urini coni

Na sliki 4.1(a) je prikazan logični primitiv negatorja, realiziran v eni urini coni. Slika 4.1(b) prikazuje rezultate simulacije njegovega delovanja. Vhodni priključek x_1 se nahaja na levem robu urine cone, izhodni priključek y_1 pa na njenem desnem robu. Širina mreže, na kateri se nahaja negator, je 6 področij, zato mora imeti urina cona z negatorjem širino najmanj 6 področij.



Slika 4.1 Logični primitiv negator, realiziran v eni urini coni (a). Vhod je označen z x_1 in izhod z y_1, pri čemer velja $y_1 = \overline{x_1}$. Priloženi rezultati simulacije potrjujejo pravilno delovanje (b).

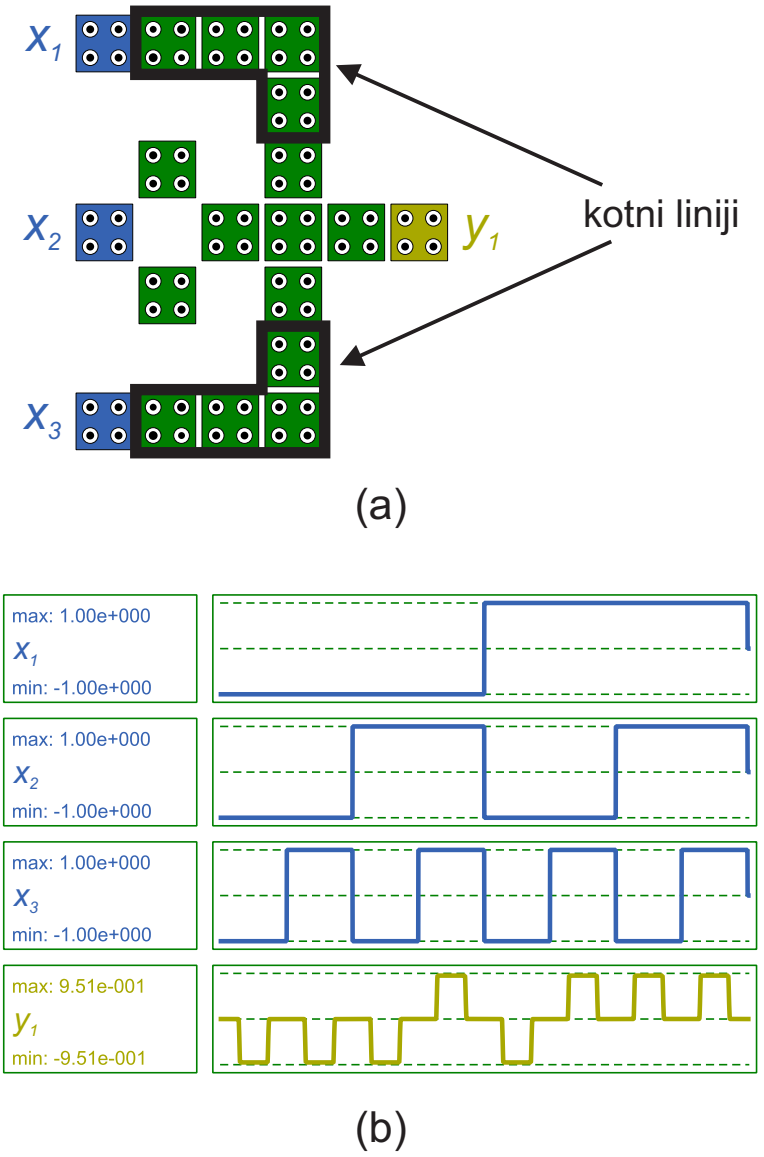
Originalna majoritetna vrata nimajo vseh vhodov na levem robu urine cone. Zato je potrebno povezati vsakega od vhodov x_1 in x_3 z levim robom cone. To se izvede z uporabo kotnih linij, kot je prikazano na sliki 4.2(a). Izkaže se, da takšen primitiv ne izvaja majoritetne funkcije. Problem se pojavi zaradi neenakih dolžin linij, ki vodijo od vhodov na levi strani do celice v središču majoritetnih vrat. Vodoravna linija med vhodom x_2 in celico v središču vsebuje dve QCA celici, linija med vhodom x_1 in središčno celico vsebuje pet QCA celic, prav tako pet QCA celic vsebuje tudi linija med vhodom x_3 in celico v središču majoritetnih vrat. Ko logični signali potujejo od vhodnih celic proti središčni celici, na slednjo vpliva le signal iz bližnjega vhoda x_2, ne pa tudi signala



Slika 4.2 Majoritetna vrata z vhodi x_1, x_2 in x_3 ter izhodom *M nepravilno* (a). Vsi vhodi se nahajajo na levi strani, vendar prikazani logični primitiv ne izvaja želene majoritetne funkcije. Izhod *M nepravilno* je vedno enak srednjemu vhodu x_2, kot prikazujejo rezultati simulacije (b).

iz bolj oddaljenih vhodov x_1 in x_3. Središčna celica se polarizira enako kot vhodna celica x_2, enako pa se nato polarizira tudi izhodna celica majoritetnih vrat.

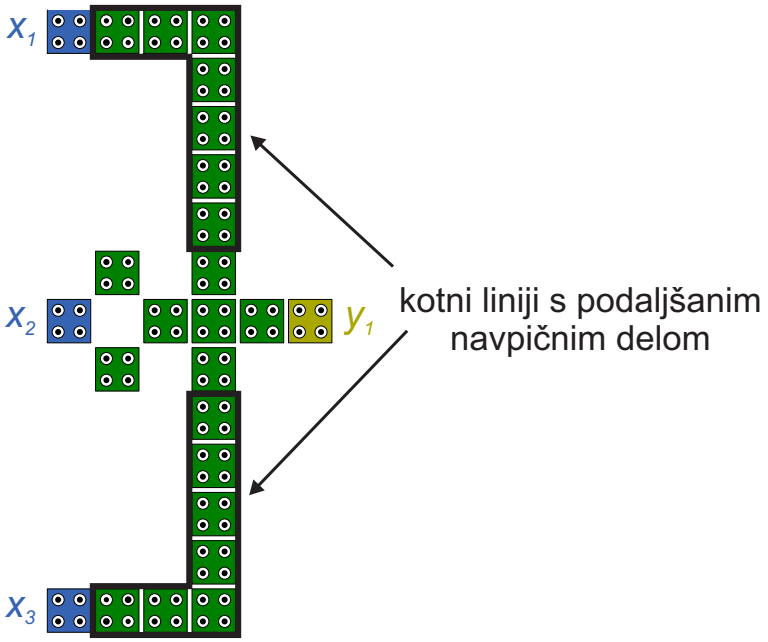
Za odpravo te nepravilnosti smo zasnovali nadgrajena majoritetna vrata, prikazana na sliki 4.3(a). Vsi vhodni signali prihajajo z leve strani, izhodni signal pa se pojavi na



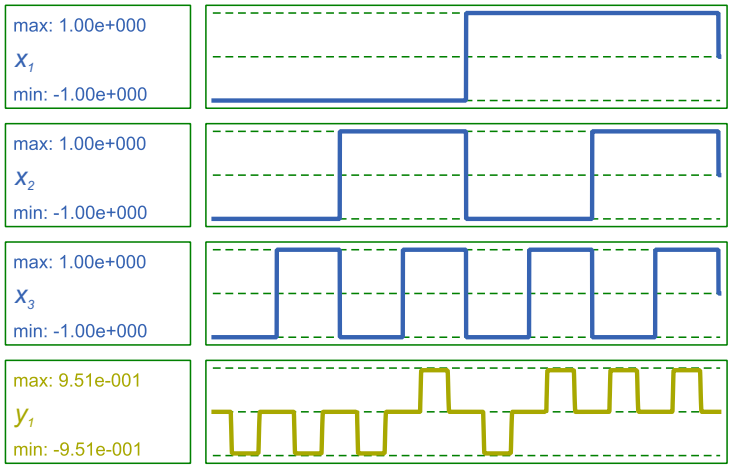
Slika 4.3 Nadgrajena majoritetna vrata z vhodi x_1, x_2 in x_3 ter izhodom y_1 (a). Logični primitiv izvaja majoritetno funkcijo $y_1 = M(x_1, x_2, x_3)$, kot prikazujejo rezultati simulacije (b).

desni strani logičnega primitiva. Dolžina linije od zgornjega vhodnega priključka x_1 do majoritetnih vrat mora biti enaka dolžini linije od spodnjega vhodnega priključka x_3 do majoritetnih vrat. Na sliki 4.3(a) tako linija od priključka x_1 kot tudi linija od priključka x_3 vsebuje dve QCA celici v vodoravnem delu linije, eno celico v kotu in eno celico v navpičnem delu linije. Logično pravilno delujejo tudi majoritetna vrata, kjer je navpični

del linije daljši, vendar mora imeti linija od priključka x_1 enako dolžino kot linija od priključka x_3. Navpična dela kotnih linij se lahko podaljšata za največ tri celice, kot je prikazano v primeru na sliki 4.4(a).



(a)

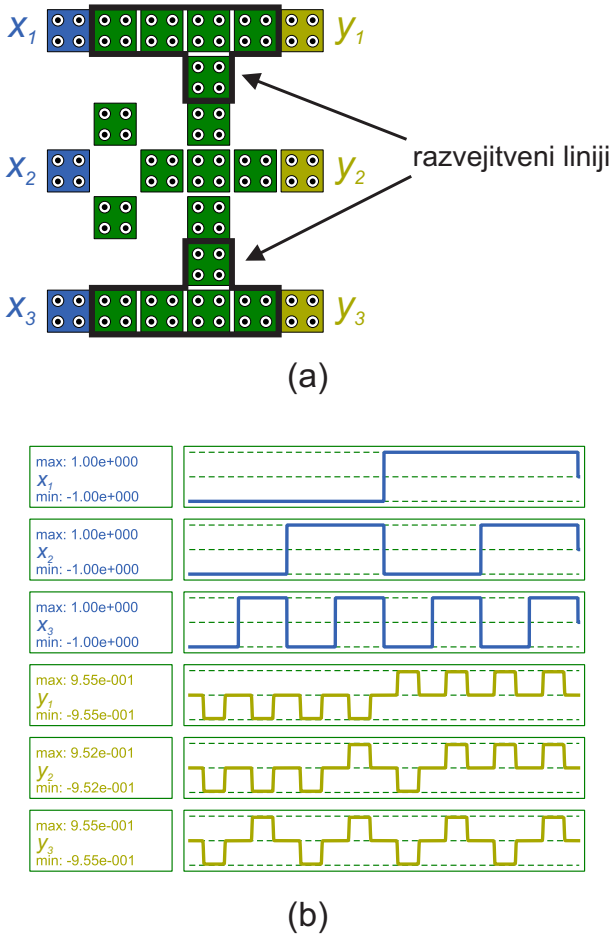


(b)

Slika 4.4 Nadgrajena majoritetna vrata s podaljšanima navpičnima deloma obeh kotnih linij (a). Priloženi rezultati simulacije potrjujejo pravilno delovanje (b).

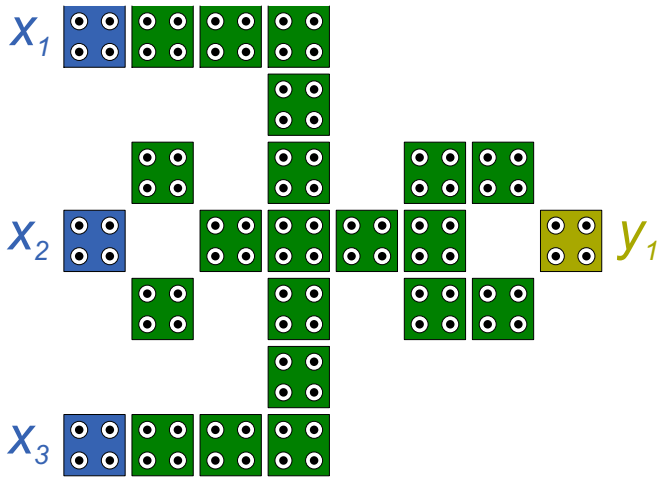
Posebno obliko ima povezava, ki poteka od vhodnega priključka x_2 do priključka na levi strani majoritetnih vrat. Področje, na katerem bi se nahajala QCA celica v ravni liniji med priključkoma, je prazno, nad in pod njim pa sta nameščeni dve celici. Levo od praznega področja se nahaja vhodni priključek x_2 na levem robu urine cone. Širina mreže s predstavljenimi majoritetnimi vrati je 6 področij.

Če se signal na vhodnem priključku x_1 potrebuje v nadaljnjem procesiranju, se vodoravna linija od priključka x_1 nadaljuje do desnega roba urine cone. Tako ni potrebna razvejitev linije v urini coni pred cono z majoritetnimi vrati. Enako pravilo velja za signal na vhodnem priključku x_3 in za oba signala na priključkih x_1 in x_3 skupaj. Slednji primer je prikazan na sliki 4.5(a).

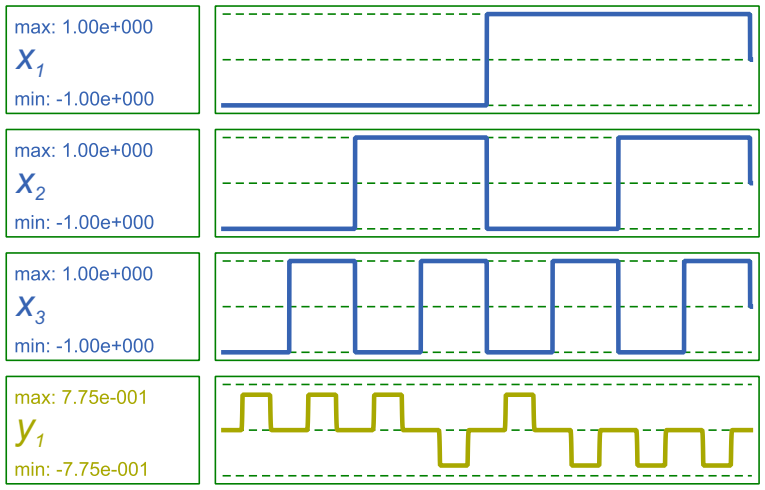


Slika 4.5 Nadgrajena majoritetna vrata z vhodi x_1, x_2 in x_3 ter izhodi y_1, y_2 in y_3 (a). Za izhode velja $y_1 = x_1$, $y_2 = M(x_1, x_2, x_3)$ in $y_3 = x_3$, kot prikazujejo priloženi rezultati simulacije (b).

Za realizacijo logičnih operacij NAND in NOR zadostuje primitiv, ki realizira negacijo majoritetne funkcije. Takšen logični primitiv, ki se v celoti nahaja v eni urini coni, je prikazan na sliki 4.6(a). Sestavljata ga stikajoča se majoritetna vrata in negator. Če



(a)



(b)

Slika 4.6 Logični primitiv, ki realizira negacijo majoritetne funkcije, torej $y_1 = \overline{M(x_1, x_2, x_3)}$ (a). Pravilno delovanje potrjujejo priloženi rezultati simulacije (b).

ima eden od vhodnih priključkov fiksno stanje z logično vrednostjo 0, ta primitiv izvaja operacijo NAND. V primeru, da je eno od stanj vhodnih priključkov fiksirano na logično

vrednost 1, primitiv izvaja operacijo NOR. S tem predstavljeni primitiv realizira operaciji iz polnega funkcijskega nabora {NAND} in {NOR} in je torej poleg linij zadosten za sestavo poljubne logične strukture. Njegova slabost v primerjavi s predhodno predstavljenimi majoritetnimi vrati je širina pripadajoče mreže, ki znaša 8 področij. Zato mora tudi širina urine cone, ki vsebuje ta primitiv, znašati vsaj 8 področij.

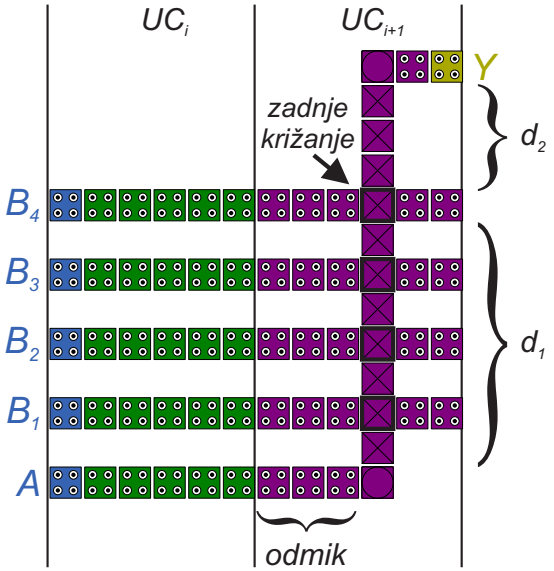
Podobno kot pri majoritetnih vratih s slike 4.5(a), je tudi v primeru logičnega primitiva negacije majoritetne funkcije možno realizirati razvejitev linije znotraj iste urine cone. Tako se lahko razvejita liniji, ki nosita signala s priključkov x_1 in x_3.

4.4 Načrtovalska pravila za realizacijo križanja linij

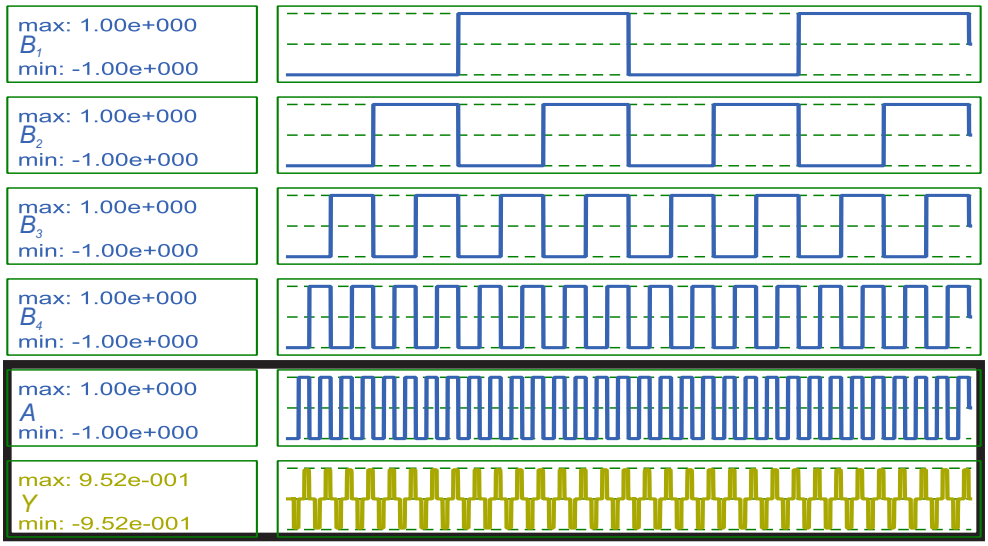
Načrtovalska pravila za realizacijo križanja linij bomo predstavili na primeru urinih con s širino 6 področij ($W(UC) = 6$). V takšne cone je možno namestiti negator ali nadgrajena majoritetna vrata. Pri področjih z dolžino stranice 20 nm znaša širina urine cone 120 nm. Vezje za generiranje urinega signala je možno izdelati, če vsota premera elektrode in razdalje med dvema elektrodama ne presega 120 nm, kar bi lahko bilo izvedljivo glede na smernice razvoja tehnologije [2]. Število QCA celic v liniji je navzgor omejeno z $N_{max} = e^{E_k/k_bT}$. Z vrednostjo $E_k = 2{,}28$ meV [75] bi linija s šestimi celicami delovala pri temperaturi pod 15 K. Vodoravne linije tako niso predolge, prevelika dolžina pa se lahko pojavi pri kotnih linijah z velikim številom celic v njihovem navpičnem delu. Zato smo dolžino navpičnega dela kotne linije omejili na 20 celic. Predolge linije se morajo razdeliti med več zaporednih urinih con.

Predstavili bomo parametre pri večnivojskem križanju. Če njihove vrednosti niso v skladu z načrtovalskimi pravili, je za zagotovitev pravilnega delovanja potrebno razdeliti kotno linijo, katere navpični del se križa z vodoravnimi linijami.

Slika 4.7(a) prikazuje primer križanja linij. Linije A, B_1, B_2, B_3 in B_4 se nahajajo v zaporednih conah UC_i in UC_{i+1}. Vse linije v UC_i so vodoravne. Linije B_1, B_2, B_3 in B_4 so vodoravne tudi v UC_{i+1}, linija A v UC_{i+1} pa ima dva kota. Linija A je v UC_{i+1} sestavljena iz začetnega dela na spodnji plasti (spodnji levi kot cone UC_{i+1} na sliki 4.7(a)), navpičnega dela na zgornji plasti in končnega dela na spodnji plasti (zgornji desni kot UC_{i+1}). Vedno se križata vodoravni del linije in navpični del neke druge kotne linije. Na sliki 4.7(a) so označena štiri križanja navpičnega dela linije A s preostalimi vodoravnimi linijami. Navpični del linije A poteka na zgornji plasti nad linijami B_1, B_2,



(a)



(b)

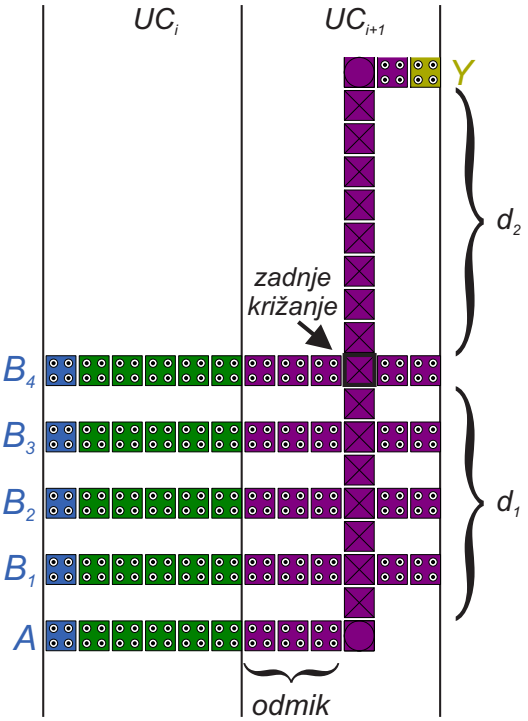
Slika 4.7 Linije A, B_1, B_2, B_3 in B_4 v conah UC_i in UC_{i+1} (a). Označena so križanja navpičnega dela linije A in preostalih vodoravnih linij v coni UC_{i+1}. Posebej je označeno zadnje križanje. Vrednosti parametrov so $odmik$=3, d_1=7 in d_2=3. Rezultati simulacije potrjujejo pravilnost delovanja (b).

B_3 in B_4 na spodnji plasti.

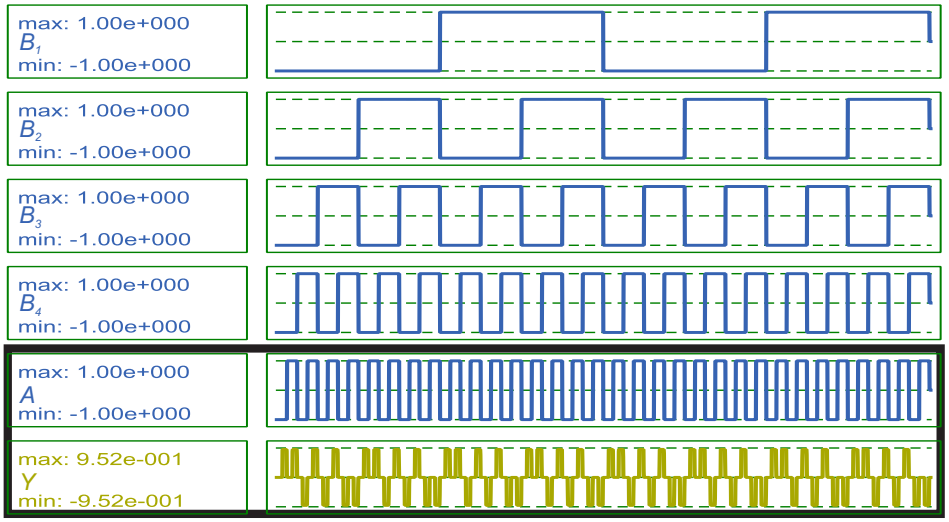
Linija na zgornji plasti se lahko križa z eno ali več linijami na spodnji plasti. Vedno obstaja zadnje križanje glede na smer poteka signala po navpičnem delu. Na sliki 4.7(a) navpični del linije A poteka od spodnjega dela UC_{i+1} proti zgornjemu delu, zato je zadnje križanje tisto z linijo B_4. Segment navpičnega dela A med začetnim delom linije A in zadnjim križanjem je označen z d_1. Ta segment se lahko križa z vodoravnimi linijami na spodnji plasti. Pri segmentu d_2 med zadnjim križanjem in končnim delom A ni nobenega križanja. Parameter *odmik* določa razdaljo med levim robom urine cone s križanji in navpičnim delom na zgornji plasti. Vsi parametri se merijo s številom področij. Vrednosti parametrov na sliki 4.7(a) so *odmik*=3, d_1=7 in d_2=3. Priloženi rezultati simulacije na sliki 4.7(b) potrjujejo pravilnost delovanja.

Slika 4.8(a) prikazuje strukturo s slike 4.7(a) s podaljšanim navpičnim delom linije A. Segment d_1=7 je enak kot na sliki 4.7(a), d_2=8 pa je podaljšan za 5 področij. Rezultati simulacije na sliki 4.8(b) prikazujejo nepravilno delovanje, saj izhod Y ni enak vhodu A. Iz obeh slik je razvidno, da je trojček *odmik*=3, d_1=7, d_2=3 (slika 4.7(a)) v skladu z načrtovalskimi pravili, trojček *odmik*=3, d_1=7, d_2=8 (slika 4.8(a)) pa ne zadošča pravilom.

Načrtovalska pravila morajo torej določiti razmerja med parametri *odmik*, d_1 in d_2, tako da bo struktura delovala pravilno. V coni s širino 6 področij je šest možnih vrednosti za parameter *odmik*, ki se nahajajo na intervalu [0, 5]. Pri izbrani vrednosti parametra *odmik* lahko d_1 zasede vrednosti med 1 in d_{1max}, torej velja $1 \leq d_1 \leq d_{1max}$. Spodnja meja 1 je potrebna zaradi minimalne razdalje med linijami. Vrednost d_{1max} je odvisna od izbrane vrednosti parametra *odmik*. Pri določenih vrednostih $odmik = x_1$ in $d_1 = x_2$ lahko d_2 zavzame poljubno vrednost na intervalu $[d_{2min}, d_{2max}]$. S simulacijo struktur z različnimi vrednostmi parametrov *odmik*, d_1 in d_2 smo definirali načrtovalska pravila, ki določajo ustrezna razmerja med parametri, pri katerih struktura deluje pravilno. Rezultati so predstavljeni z grafi na sliki 4.9. Šest grafov ustreza šestim možnim vrednostim parametra *odmik*. Vsak graf prikazuje minimalno (d_{2min}, označen s križci) in maksimalno (d_{2max}, označen s krogci) dovoljeno vrednost d_2 pri določenem d_1. Pri danem d_1 so vse vrednosti med minimalno in maksimalno dopustne za d_2. Vsaka točka (x, y) znotraj ali na robu osenčenega lika pri $odmik = z$ na sliki 4.9 določa veljaven trojček $odmik = z$, $d_1 = x$, $d_2 = y$, če so x, y, z naravna števila. Na grafu z $odmik = 3$ točka d_1=7, d_2=3 leži na robu osenčenega lika, kar pomeni da so parametri pri strukturi

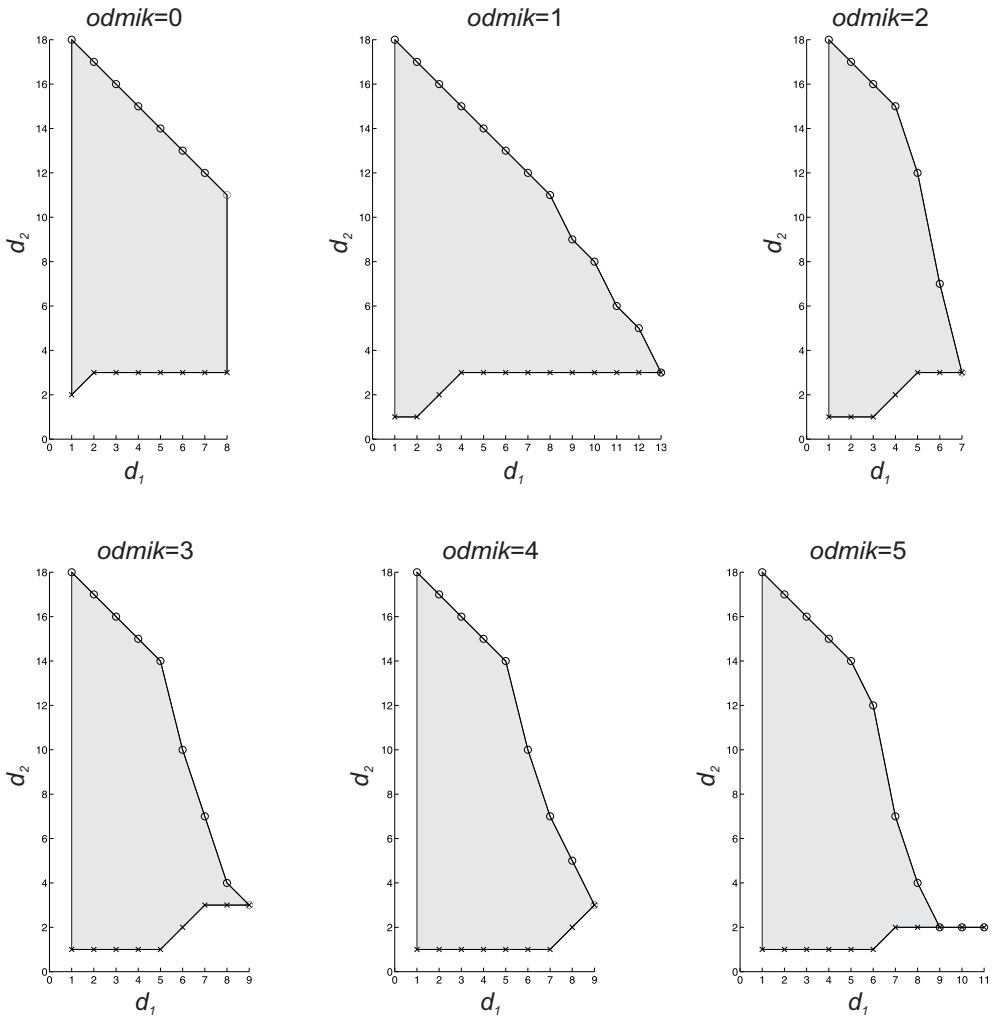


(a)



(b)

Slika 4.8 Struktura z vrednostmi parametrov $odmik$=3, d_1=7, d_2=8 (a). Rezultati simulacije kažejo, da parametri niso v skladu z načrtovalskimi pravili (b).



Slika 4.9 Načrtovalska pravila določajo dovoljena razmerja med parametri d_1, d_2 in $odmik$. Šest grafov zajema vse možnosti v urini coni s širino 6 področij. Vsak graf določa minimalno ($\times$) in maksimalno ($\circ$) dovoljeno dolžino d_2 glede na pripadajočo dolžino d_1. Vsaka točka (x, y) znotraj ali na robu osenčenega lika pri $odmik = z$ določa veljaven trojček $odmik = z$, $d_1 = x$, $d_2 = y$, če so x, y, z naravna števila. Struktura z veljavnim trojčkom parametrov je zasnovana v skladu z načrtovalskimi pravili.

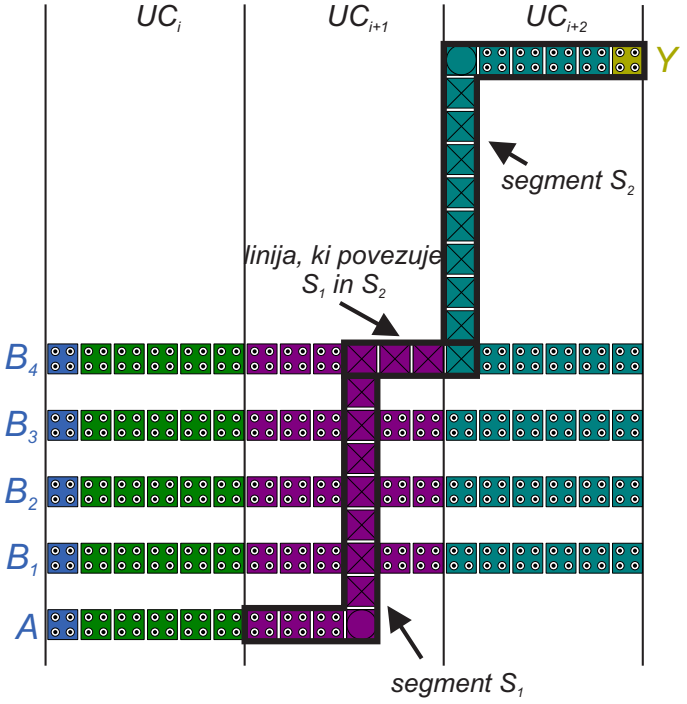
na sliki 4.7(a) veljavni. Nasprotno pa točka d_1=7, d_2=8 leži zunaj osenčenega lika, zato je trojček *odmik*=3, d_1=7, d_2=8 neveljaven.

Za odpravo problema pri strukturi na sliki 4.8(a) se mora kotna linija razdeliti med dve zaporedni coni. Najprej se določi celica C na navpičnem delu kotne linije znotraj cone UC_j. Kotna linija se razdeli v segmenta S_1 in S_2. Segment S_1 leži med vhodno celico kotne linije na levem robu UC_j in celico C, segment S_2 pa med C in izhodno celico kotne linije na desnem robu UC_j. Nato se vstavi nova vmesna urina cona UC_{j+1} desno od UC_j in segment S_2 se prenese iz UC_j v UC_{j+1}. Nazadnje se ločena segmenta S_1 in S_2 povežeta z vodoravno linijo, nameščeno na zgornjo plast. Z razdelitvijo kotne linije s slike 4.8(a) na opisan način dobimo strukturo na sliki 4.10(a). Za celico C, ki deli kotno linijo na segmenta S_1 in S_2, je določena celica na mestu zadnjega križanja v coni UC_{i+1}. Dodana je cona UC_{i+2}, v katero se iz cone UC_{i+1} prestavi segment S_2. Segmenta sta nato povezana z linijo, nameščeno na zgornji plasti, ki poteka nad linijo B_4 na spodnji plasti.

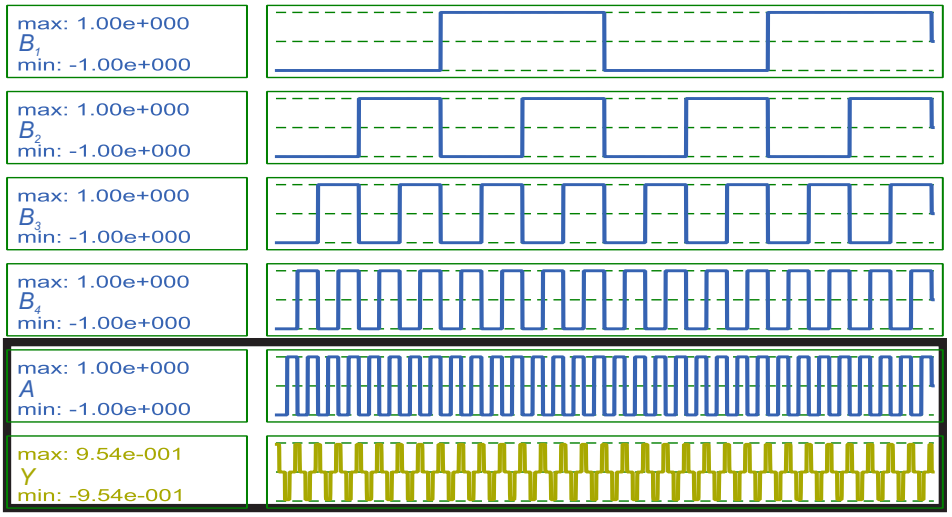
Kotna linija, ki leži v coni UC_{i+1} na sliki 4.10(a), je sedaj sestavljena iz segmenta S_2 in začetnega dela linije med S_1 in S_2. Kotna linija se križa z linijami B_1, B_2 in B_3, pri čemer je križanje z B_3 sedaj zadnje križanje. Parametri pri tej kotni liniji so *odmik*=3, d_1=5 in d_2=1. S slike 4.9 je razvidno, da parametri sestavljajo veljaven trojček. Če parametri ne bi zadoščali načrtovalskim pravilom, bi se morala vstaviti nova cona med UC_{i+1} in UC_{i+2}. Kotna linija bi se nato po opisanem postopku razdelila med UC_{i+1} in novo vstavljeno cono. Postopek se izvede za vsako kotno linijo v fizični razmestitvi strukture, dokler niso vsi parametri v skladu z načrtovalskimi pravili. Kotno linijo v UC_{i+2} na sliki 4.10(a) sestavljata segment S_2 in najbolj desna celica v liniji med S_1 in S_2. Ker se ne križa z nobeno drugo linijo, pravila s slike 4.9 ne pridejo v poštev. Dolžina njenega navpičnega dela ne presega zgornje meje 20 celic, zato kotne linije ni potrebno nadalje razdeliti.

4.5 Minimalna potrebna razdalja med različnimi logičnimi primitivi

Simulacije kažejo, da mora biti med različnimi primitivi vsaj eno prosto področje. To pomeni, da mora biti vsaj eno prosto področje med vzporednimi linijami, ali med različnimi logičnimi vrati, ali med vrati in linijo, ki nista povezana. Kadar potrebne razdalje



(a)



(b)

Slika 4.10 Kotna linija s slike 4.8(a) je razdeljena na segmenta S_1 v UC_{i+1} in S_2 v UC_{i+2}. Segmenta sta povezana z linijo, nameščeno na zgornji plasti, ki poteka nad linijo B_4 na spodnji plasti (a). Rezultati simulacije izkazujejo pravilno delovanje (b).

ni, pride do interakcije med sosednjimi celicami, ki pa sestavljajo različna primitiva. To privede do nepravilnega delovanja primitivov. Prazen prostor med različnimi primitivi prepreči takšne neželene interakcije med celicami.

Energija vzbujanja E_k je obratno sorazmerna s peto potenco razdalje med celicama [20], zato se hitro zmanjša z majhnim povečanjem razdalje. Zato je eno prazno področje med različnimi primitivi dovolj velika razdalja za preprečitev neželenih interakcij.

5 Metrike

5.1 Uvod

Za potrebe ovrednotenja lastnosti računalniške strukture je potrebno določiti ustrezne metrike. Glede na vpeljane metrike je mogoče med seboj primerjati različne zasnove struktur. Potrebno je določiti tudi osnovne merske enote. Metrike so odvisne od tehnologije izdelave strukture. Pri različnih tehnologijah se zato razlikujejo tudi merske enote.

Ker je QCA v celoti sestavljen iz celic, je za osnovno mersko enoto primerna ena QCA celica. Njene dimenzije so odvisne od tehnologije realizacije. Pri QCA je pomembno minimizirati število kotnih linij in število križanj linij, ker so težje izvedljivi, napake pri izdelavi pa lahko povzročijo napačno delovanje celotne strukture. Zaradi specifičnih načrtovalskih pravil so parametri pri križanju linij pomembni za določanje velikosti strukture in števila potrebnih urinih faz za izračun rezultata. Za razliko od CMOS tehnologije, kjer je križanje žic možno na večjem številu nivojev, gre pri QCA pričakovati le možnost dvonivojskega križanja [32]. Zato je minimizacija števila križanj linij v QCA tehnologiji zelo pomembna.

Fizično razmestitev strukture QCA sestavljajo med seboj ustrezno povezani logični primitivi, ki realizirajo logične operacije v izbranem funkcijskem naboru. Če operacije pripadajo polnemu funkcijskemu naboru, je na ta način možno realizirati poljubno logično funkcijo. Množico vseh N_1 primitivov v fizični razmestitvi strukture označimo s P=$\{p_1, p_2, ..., p_{N_1}\}$. Razdelimo jo lahko v dve disjunktni podmnožici $\mathrm{P}_{operatorji}$ in P_{linije}. V $\mathrm{P}_{operatorji}$=$\{po_1, po_2, ..., po_{N_{operatorji}}\}$ se nahajajo logični primitivi, ki realizirajo logične operacije. Privzeli bomo, da so elementi množice $\mathrm{P}_{operatorji}$ negatorji in majoritetna vrata. Elementi množice P_{linije}=$\{pl_1, pl_2, ..., pl_{N_{linije}}\}$ so primitivi, ki realizirajo povezave v QCA. Ti primitivi so torej navadne in 45 stopinjske linije, realizacije prehoda signala med njima, kotne in razvejitvene linije ter realizacije križanj linij.

Število vseh primitivov je določeno z vsoto

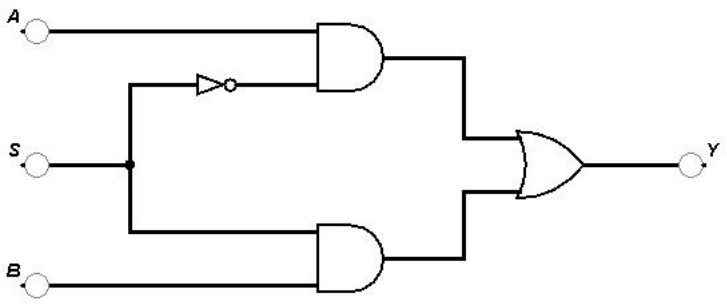
$$N_1 = N_{operatorji} + N_{linije}. \tag{5.1}$$

Število $N_{operatorji}$ elementov množice $\mathrm{P}_{operatorji}$ je določeno z logično shemo in je enako v vseh fizičnih razmestitvah strukture, zasnovanih na osnovi iste logične sheme. Med različnimi fizičnimi razmestitvami strukture pa se razlikuje število linij N_{linije}.

Kot zgled bomo v nadaljevanju obravnavali enostavno logično funkcijo enonaslovnega multiplekserja $2/1\_mux(A, B, S) = Y$, podano z izrazom

$$Y = A\overline{S} \vee BS. \tag{5.2}$$

Pripadajoča logična shema je prikazana na sliki 5.1. Na podlagi te logične sheme lahko

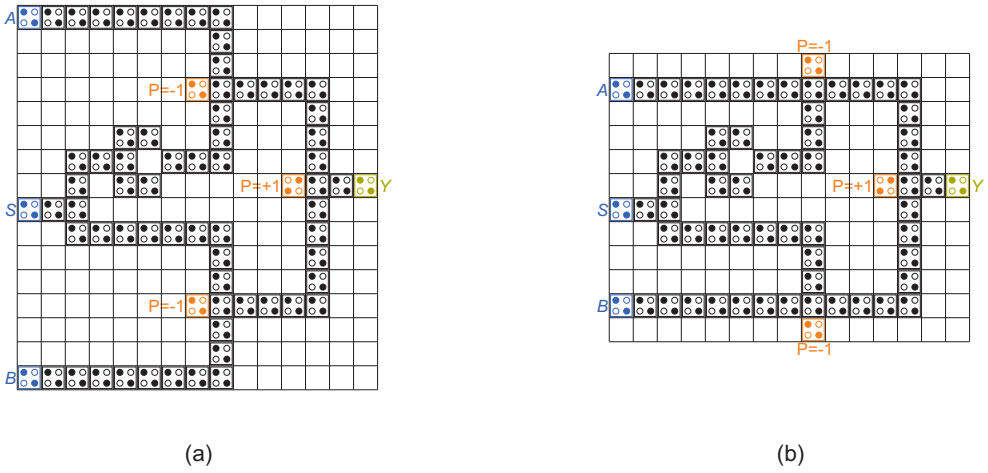


Slika 5.1 Logična shema enonaslovnega multiplekserja $2/1\_mux(A, B, S) = Y$ z vhodnimi priključki A, B, S in izhodnim priključkom Y.

z metodo fizičnega snovanja sestavimo različne fizične razmestitve strukture QCA. Z $v(s)$ označimo logično vrednost signala na priključku in zapišemo kombinacijo logičnih

vrednosti na vhodih z $v(A) = x_1$, $v(B) = x_2$, $v(S) = x_3$ in vrednost na izhodu z $v(Y) = x_4$. Če sta dve fizični razmestitvi strukture QCA zasnovani na podlagi iste logične sheme in je na njunih vhodih enaka kombinacija logičnih vrednosti, potem je tudi vrednost njunih izhodov enaka. Razlika je v tem, da lahko eno od fizičnih razmestitev strukture sestavljajo daljše linije, ima večjo zakasnitev signala, itd.

Na sliki 5.2 sta prikazani dve različni strukturi QCA. Obe sta zasnovani na podlagi logične sheme s slike 5.1 z ad hoc postopkom. Obe vsebujeta en negator in tri majoritetna vrata ($N_{operatorji}$=4), struktura na sliki 5.2(b) pa vsebuje dve kotni liniji manj kot struktura na sliki 5.2(a).



Slika 5.2 Dve različni fizični razmestitvi strukture QCA, ki realizirata logično funkcijo enonaslovnega multiplekserja $2/1\_mux(A, B, S) = Y$. Obe sta zasnovani na podlagi logične sheme s slike 5.1.

5.2 Število potrebnih urinih faz za izračun rezultata

Procesiranje v fizični razmestitvi strukture QCA je računanje rezultata logične funkcije. Prične se s prihodom vhodnih signalov v vhodne celice in se zaključi, ko so izhodni signali prisotni v izhodnih celicah. Logične vrednosti izhodnih signalov predstavljajo rezultat procesiranja. Izhodni signal z indeksom i je na voljo po zakasnitvi d_i, merjeni v številu urinih faz. Število potrebnih urinih faz za izračun rezultata d je enako največji izmed vseh zakasnitev d_i. Zakasnitev d je torej enaka številu vseh urinih con, preko katerih poteka signal od vhodne celice strukture do izhodne celice strukture.

5.3 Vsota dolžin vseh linij

Število vhodnih celic strukture je enako številu vhodov N_{vhodi}, število izhodnih celic strukture pa je enako številu izhodov N_{izhodi}. Število celic v primitivu p_i označimo z $N_{celice}(p_i)$. Tako je število vseh QCA celic v strukturi N_{celic} enako vsoti

$$N_{celic} = N_{vhodi} + N_{izhodi} + \sum_{i=1}^{N_1} N_{celice}(p_i). \tag{5.3}$$

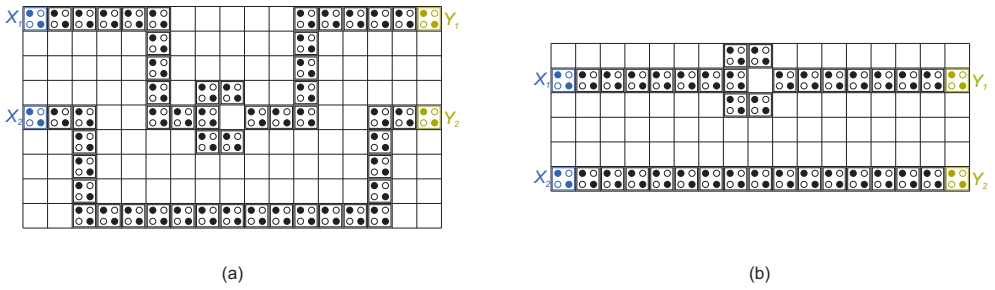
Na podlagi enačbe (5.1) lahko vsoto (5.3) zapišemo kot

$$N_{celic} = N_{vhodi} + N_{izhodi} + \sum_{i=1}^{N_{operatorji}} N_{celice}(po_i) + \sum_{j=1}^{N_{linije}} N_{celice}(pl_j). \tag{5.4}$$

Množica logičnih primitivov, ki realizirajo logične operatorje, je enaka v vseh fizičnih razmestitvah strukture, zasnovanih glede na isto logično shemo. Prav tako sta konstantni števili vhodov in izhodov strukture. Zato so v našem primeru prvi trije členi vsote (5.4) konstantni. Tako v našem primeru minimiziramo le vsoto QCA celic v linijah.

5.4 Število kotnih linij

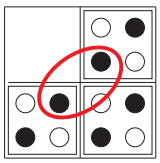
Kotna linija je potrebna povsod, kjer je potreba po spremembi toka podatkov za 90°. V postopku razmeščanja pa se lahko število kotnih linij zmanjša, kot je prikazano na sliki 5.3. Obe fizični razmestitvi strukture QCA realizirata isto logično funkcijo, vendar



Slika 5.3 Dve različni fizični razmestitvi strukture QCA, ki realizirata isto logično funkcijo. Struktura na sliki (a) vsebuje štiri kotne linije, struktura na sliki (b) pa nobene.

struktura na sliki 5.3(a) vsebuje osem kotnih linij, struktura na sliki 5.3(b) pa nobene, kar je bilo doseženo z drugačnim razmeščanjem. S tem struktura vsebuje tudi manjše število QCA celic in manjšo vsoto dolžin vseh linij.

V kotni liniji se lahko pojavi konfliktna lega elektronov v celicah na diagonali [97], kot je prikazano na sliki 5.4. Takšna lega elektronov izkazuje manj zanesljivo delovanje kot lega elektronov v ravni liniji. Kotna linija je tudi bolj kot ravna linija občutljiva na napake pri izdelavi QCA. Takšna napaka je na primer manjkajoča celica v liniji. Simulacije so pokazale, da ravna linija še vedno izvaja funkcijo identitete, čeprav v njej manjka ena celica [98]. Če pa v kotni liniji manjka QCA celica v kotu, imata celici na diagonali nasprotni polarizaciji. Tedaj se logična vrednost signala na kotni liniji negira, namesto da bi ostala nespremenjena.



Slika 5.4 Kotna linija z označeno konfliktno lego elektronov v celicah na diagonali.

5.5 Porabljena površina

Porabljeno površino določa minimalna pravokotna mreža, ki še vsebuje fizično razmestitev strukture. Mreža ima širino w in višino h. Enota za merjenje njunih dolžin je posamezno področje v mreži, na katerem je lahko nameščena QCA celica.

Mreža m_1 na sliki 5.3(a) ima širino $w = 17$ področij in višino $h = 9$ področij. Porabljena površina $A(m_1)$ je enaka velikosti te mreže, torej $A(m_1) = 17 \times 9 = 153$ področij. V primeru na sliki 5.3(b) ima mreža m_2 s širino $w = 17$ področij in višino $h = 6$ področij velikost $A(m_2) = 17 \times 6 = 102$. Glede na kriterij porabljene površine je torej mreža m_2 optimalnejša od mreže m_1.

5.6 Število križanj linij

Številne kompleksne logične sheme vsebujejo križanja povezav. Število križanj je možno minimizirati z uporabo različnih algoritmov, vendar pa ni mogoče vedno odpraviti vseh. To pomeni, da se v fizični razmestitvi strukture marsikdaj križanju linij ni možno izogniti.

V CMOS vezju se dve žici, ki prenašata signal med moduli, ne moreta križati na istem nivoju. Nezadostna razdalja med žicama povzroči nezaželene električne učinke in ne omogoča prenosa signala. Problem križanja se v CMOS tehnologiji rešuje z večplastnimi

vezji. Kadar se dve žici križata, sta nameščeni na različnih plasteh vezja, med njima pa se nahaja izolator.

V QCA so povezave sestavljene izključno iz QCA celic. Ker številne strukture neizbežno vsebujejo križanja linij, je za zagotovitev uporabnosti QCA problem potrebno rešiti. Pojavilo se je več predlogov za rešitev problema. Eden izmed predlogov je rešitev z uporabo več plasti [69], ki je analogna implementaciji v CMOS vezjih, ostali pa izkoriščajo posebnosti delovanja QCA. Vsaka rešitev ima svoje prednosti in slabosti. Nekatere imajo večjo možnost praktične realizacije kot druge. Ideje za reševanje problema križanja linij v QCA so predstavljene v naslednjih alinejah:

- **Minimizacija križanja linij:** Implementacija križanja linij je prostorsko bolj potratna od implementacije običajne linije. Predvideva se, da bo križanje linij težko fizično izdelati. Če se ena linija večkrat križa z drugimi, mora biti vsako križanje implementirano v svoji urini coni, kar pomeni večjo zakasnitev signala na liniji. Zato je smiselno minimizirati število križanj linij in izračunati minimalno potrebno število križanj v obravnavani strukturi. Problem minimizacije števila križanj je NP-težak, zato raziskovalci [21, 31, 30, 85] za iskanje približne rešitve uporabljajo razne hevristične metode.

  Kljub minimizaciji je število potrebnih križanj v mnogih fizičnih razmestitvah strukture QCA večje od nič, zato je še vedno potrebno implementirati križanje linij.

- **Križanje linij na istem nivoju z uporabo rotiranih QCA celic:** Tougaw in Lent [4] sta z izračuni prikazala možnost križanja linij v QCA na istem nivoju. Takega križanja v CMOS vezjih ni mogoče implementirati, zato je ta posebnost predstavljala še eno izmed prednosti QCA pred CMOS tehnologijo. Izračuni so pokazali, da se signala na navadni in 45 stopinjski liniji pri križanju nemoteno preneseta vsak po svoji liniji.

  Sprva je možnost istonivojskega križanja linij delovala obetavno, kasnejše raziskave pa so pokazale slabosti te rešitve. Največji oviri pri implementaciji križanja navadne in 45 stopinjske linije sta majhna robustnost in kompleksna fizična izdelava. V [75] je prikazana motnja prenosa signala pri več zaporednih križanjih. Poleg tega križanje linij ne deluje zanesljivo, če je v bližini nameščena QCA celica s fiksnim stanjem. Majhna robustnost izhaja iz dejstva, da je razdalja med dvema QCA celicama v navadni liniji pri križanju večja kot v primeru 45 stopinjske linije. Drugo

oviro za uporabo istonivojskega križanja linij predstavlja težavna natančna namestitev tako majhnih elementov kot so QCA celice [25]. Zaradi tega je z obstoječo tehnologijo zelo težko fizično namestiti rotirane in zamaknjene QCA celice.

- **Večnivojsko križanje linij:** Večnivojsko križanje linij v QCA je analogno implementaciji križanja žic v CMOS tehnologiji. V primeru križanja sta liniji nameščeni na različnih nivojih. Pri tem je potrebno sestaviti več nivojev, na katere bodo nameščene QCA celice. Hkrati mora biti izdelana povezava med nivoji, tako da lahko linija preide na višje oziroma nižje ležeči nivo.

  Medtem ko je takšna implementacija križanja prisotna v CMOS tehnologiji, v QCA še ni bila eksperimentalno realizirana. Možnost njene izvedbe je predvidoma zelo odvisna od realizacije QCA. V primeru uspešne izdelave bi lahko imela večnivojska struktura QCA še eno prednost pred večnivojskim CMOS vezjem. V slednjem so dodatni nivoji namenjeni le namestitvi žic zaradi izogibanja križanju na istem nivoju, v QCA strukturi pa bi na kateremukoli nivoju lahko bili nameščeni tudi logični primitivi, ki realizirajo logične operacije, saj so sestavljeni iz enakih osnovnih gradnikov kot linije.

- **Križanje linij na istem nivoju z uporabo faznega zamika urinega signala:** Ta rešitev za križanje linij na istem nivoju izkorišča QCA uro. Avtorji so v delu [33] predlagali vpeljavo osemfazne ure s tremi tipi signalov. Preklop stanja QCA celic na območju križanja linij je nadzorovan z ustrezno fazo novega urinega signala, kar omogoči nemoten potek signalov po linijah.

  Slabost predlagane rešitve je v kompleksnejši izvedbi QCA ure in večji zakasnitvi signala v strukturi. Poleg tega je za vsako QCA celico posebej na območju križanja urin signal zamaknjen za eno fazo. To lahko predstavlja težavo za izdelavo ustreznega CMOS vezja, ki bo omogočilo urin signal QCA.

- **Logično križanje linij:** Križanje linij na istem nivoju je možno implementirati z logičnim elementom z dvema vhodoma na levi strani in dvema izhodoma na desni. Izhodna signala sta enaka vhodnima, pri čemer se na izhodu pojavita v obratnem vrstnem redu kot na vhodu. Seveda pa mora biti tak logični element realiziran brez križanja linij.

6 Algoritmi za razmeščanje in povezovanje v CMOS tehnologiji

6.1 Uvod

Namen fizičnega snovanja je definiranje fizične postavitve vezja na podlagi njegovega logičnega opisa. Slednji je podan z logično strukturo, ki jo sestavljata množica logičnih operatorjev in množica povezav med operatorji. Rezultat procesa fizičnega snovanja je fizična razmestitev strukture, sestavljena iz logičnih primitivov in fizičnih povezav med njimi. Vsak logični primitiv ustreza enemu izmed operatorjev v logični strukturi, vsaka fizična povezava pa je realizacija ene izmed povezav v logični strukturi. V fizični razmestitvi strukture so položaji logičnih primitivov in fizičnih povezav natančno določeni. Tako je fizična razmestitev strukture podlaga za končno izdelavo strukture.

Na začetku razvoja polprevodniške tehnologije so bila vezja relativno enostavna z majhnim številom gradnikov. Fizično snovanje je zato lahko potekalo ročno. Sčasoma so vezja postajala vse bolj kompleksna, število njihovih gradnikov pa je začelo eksponentno naraščati. Ročno snovanje takih vezij ni bilo več mogoče in pojavila se je avtomatizacija procesa. Slednja ne omogoča le snovanja vezij z velikim številom gradnikov, temveč tudi optimizacijo različnih kriterijev in dosledno upoštevanje postavljenih načrtovalskih

pravil.

Za avtomatsko snovanje je bilo potrebno razviti različne algoritme in jih programsko implementirati. Ti algoritmi so se razvijali in nadgrajevali hkrati z razvojem tehnologije za izdelavo integriranih vezij. Omogočajo avtomatizacijo celotnega procesa fizičnega snovanja, sestavljenega iz stopenj snovanja geometrije strukture, razmeščanja in povezovanja. V stopnji snovanja geometrije se površina za namestitev strukture razdeli na področja in logični primitivi se razdelijo v skupine, tako da vsaki skupini ustreza eno področje. Vsi primitivi v skupini se namestijo v pripadajoče področje. Pri snovanju CMOS vezja se področja in skupine določijo glede na optimizacijske kriterije, v QCA strukturi pa področje ustreza urini coni. Pri slednji je zato potrebno določiti tako geometrijo, ki bo omogočila fizično izdelavo. Poleg tega se logični primitivi razdelijo v skupine glede na medsebojne povezave, zato se snovanje geometrije CMOS vezja precej razlikuje od snovanja geometrije QCA strukture. Obstaja pa veliko podobnosti med stopnjama razmeščanja v CMOS in v QCA tehnologiji. Enako velja za povezovanje, zato bomo v tem poglavju obravnavali algoritme za razmeščanje in povezovanje v CMOS tehnologiji.

6.2 Algoritmi za razmeščanje v CMOS tehnologiji

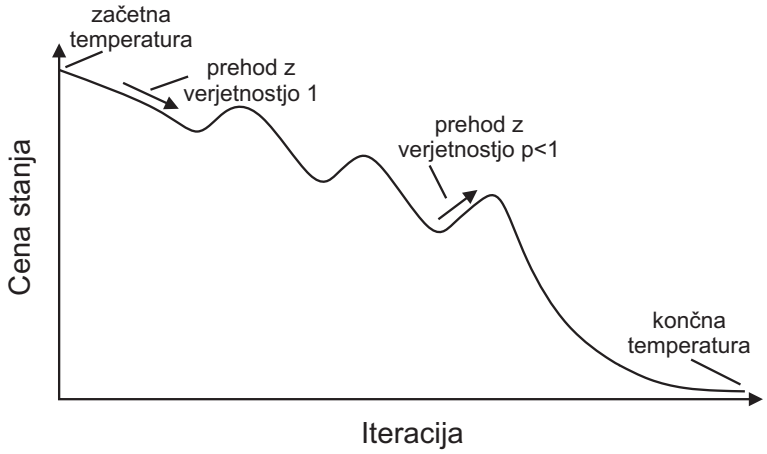
Cilj metod razmeščanja je določiti položaje gradnikov na čipu, tako da bo vezje možno fizično realizirati, pri tem pa se optimizirajo različni kriteriji. Tipični kriteriji so ocene skupne dolžine povezav, dolžine povezav na kritični poti, gostote povezav, porabe energije in ostali. Pri iskanju optimalne razmestitve in povezanosti glede na dane pogoje je večina problemov NP-težkih. Pri snovanju se zato uporabljajo temu primerne tehnike, kot so aproksimacijski in verjetnostni algoritmi ter različne hevristike. Problemi, pri katerih obstaja optimalna rešitev znotraj določenega podsistema, se lahko optimizirajo s *požrešnim algoritmom* (angl. *greedy algorithm*). Razmeščanje logičnih primitivov ne spada v omenjeno skupino problemov, zato se požrešni algoritem navadno ujame v lokalni minimum. Metode iskanja suboptimalne razmestitve obsegajo *simulirano ohlajanje* [12], *analitične metode* [13, 14] ter metode *na osnovi sil* (angl. *force-directed*) [15] in *na osnovi razdelitve* (angl. *partitioning*) [16].

6.3 Uporaba simuliranega ohlajanja za razmeščanje

Simulirano ohlajanje je ena od metod za optimizacijo *kombinatoričnih problemov* [99]. Pri slednjih ima vsaka od rešitev pripadajočo ceno, iščemo pa čim cenejšo rešitev. Za iskanje rešitve s ceno, ki je čim bližja *globalnemu minimumu*, metoda uporablja hevristike, s pomočjo katerih se iskanje ne zaustavi v *lokalnem minimumu*. Simulirano ohlajanje je opisal Kirkpatrick s sodelavci [12] leta 1983 in od takrat se uporablja za reševanje raznih problemov z diskretnimi stanji. Eden od njih je tudi razmeščanje gradnikov pri fizičnem snovanju integriranih vezij.

Ideja za algoritem izhaja iz ohlajanja materialov v metalurgiji. V začetnem stanju je material segret na visoko temperaturo, pri čemer se atomi premaknejo iz začetnega položaja in med naključnim premikanjem lahko dosežejo višja energetska stanja od začetnega. Z gibanjem atomov se odpravijo napake v materialu. Med počasnim ohlajanjem se verjetnost povečanja energije zmanjšuje. Material prehaja v ravnovesno stanje z minimalno skupno energijo.

Opisani postopek je osnova optimizacijskega algoritma simuliranega ohlajanja, podanega v psevdokodi v algoritmu 1. Njegovo delovanje je shematično prikazano na sliki 6.1. Algoritem za iskanje globalnega minimuma sistema izvaja iterativno izboljševanje.



Slika 6.1 Shema delovanja algoritma simuliranega ohlajanja.

Najprej se določi začetno stanje sistema in visoka začetna temperatura. Nato se prične izvajati zanka, kjer se v vsaki iteraciji izbere novo stanje *snew* in izračuna njegova cena *cnew*. Novo stanje *snew* se izbere naključno na podlagi vnaprej definiranega sosedstva

Algoritem 1 Simulirano ohlajanje.

```
procedure SIMULIRANO_OHLAJANJE
// določi začetno stanje in njegovo ceno
    s = s_0; c = cena(s)
// prva najboljša rešitev je začetno stanje
    sbest = s; cbest = c
// določi začetno temperaturo
    t = t_0
    while t > 0 do
      // izberi novo trenutno stanje
        snew = sosed(s); cnew = cena(snew)
      // ali je novo stanje najboljša rešitev
        if cnew < cbest then
            sbest = snew; cbest = cnew
        end if
      // izračunaj verjetnost prehoda v novo stanje glede na trenutno temperaturo
      // če je cnew < c, potem p = 1; sicer se p manjša z nižanjem temperature
        p = verjetnost_prehoda(c, cnew, t)
      // prehod v novo stanje
      // r ∈ [0,1) je naključno izbrano število
        if p > r then
            s = snew; c = cnew
        end if
      // zmanjšaj temperaturo s funkcijo ohlajanja
        t = ohlajanje(t)
    end while
return sbest
end procedure
```

glede na trenutno stanje s. Če ima izbrano novo stanje *snew* ceno, ki je nižja od cene trenutnega stanja s, potem novo trenutno stanje postane stanje *snew*. Tudi če ima stanje *snew* višjo ceno od cene trenutnega stanja s, obstaja možnost izbire *snew* za novo trenutno stanje. Tako obstaja manj možnosti, da bi se iskanje zaustavilo v lokalnem

minimumu. Verjetnost izbire stanja *snew* z višjo ceno od trenutne je višja pri visoki temperaturi in se med ohlajanjem niža. V vsaki iteraciji se temperatura zniža glede na funkcijo ohlajanja. Izvajanje algoritma se zaključi, ko se doseže končna temperatura.

Zaradi možnosti prehoda v stanje z višjo ceno lahko iskanje uide iz lokalnega minimuma. Verjetnost takega prehoda se med izvajanjem niža, tako da se sistem sčasoma ustali v globalnem minimumu oziroma v njegovem približku. Za čim boljši približek je potrebno preiskati čim večji del prostora stanj. Ker je ta prostor pri kompleksnih problemih zelo velik, je lahko iskanje globalnega minimuma dolgotrajno. Simulirano ohlajanje je splošna hevristična metoda, zato se lahko uporablja na različnih področjih. Pri razmeščanju je z njo možno optimizirati različne kriterije in tudi več kriterijev hkrati.

6.4 Algoritmi za povezovanje v CMOS tehnologiji

Metode povezovanja ob upoštevanju načrtovalskih pravil določijo lege povezav med ustreznimi gradniki. Najpomembneje pri tem je zagotoviti prostor za povezave, tako da bodo lahko nameščene vse povezave in bo vezje delovalo pravilno. Poleg tega se tudi optimizirajo izbrani kriteriji, kot je na primer skupna dolžina povezav.

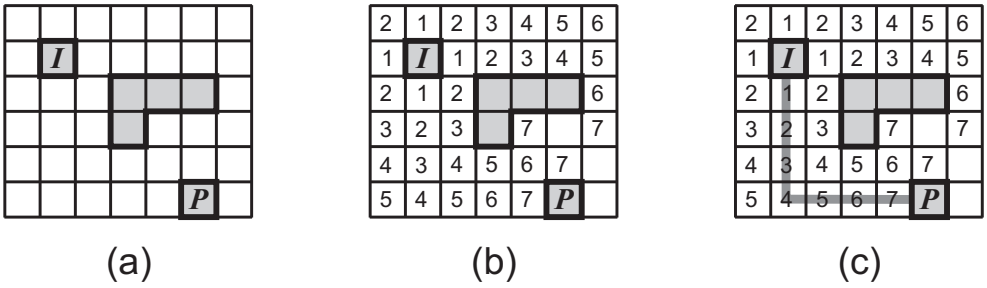
Povezovanje v kompleksnem vezju, ki vsebuje veliko število povezav, je zaradi obvladovanja težavnosti problema običajno razdeljeno na dve stopnji: *globalno povezovanje* (angl. *global routing*) in *podrobno povezovanje* (angl. *detailed routing*). V stopnji globalnega povezovanja se v grobem določi prostor za namestitev povezave. Površina za namestitev povezav je razdeljena na področja. Za vsako povezavo se določi področja, kamor se bo povezava namestila. Če so področja zadosti velika, je površina razdeljena na dovolj majhno število področij, tako da prostor stanj ni prevelik. Takšen problem se lahko rešuje z algoritmi, ki poiščejo optimalno rešitev, vendar so časovno in prostorsko zahtevni. Ko so področja izbrana, se v stopnji podrobnega povezovanja določijo natančne lege povezav znotraj področij. Pri tem je prostor stanj v kompleksnem vezju zelo velik, zato se za podrobno povezovanje običajno uporabljajo hitri algoritmi, ki poiščejo suboptimalno rešitev.

Za povezovanje gradnikov se uporabljajo tri glavne strategije in njihove kombinacije. Uporabljane strategije so *iskanje v širino* (angl. *breadth-first*), *iskanje v globino* (angl. *depth-first*) in iskanje po načelu *najprej najboljši* (angl. *best-first*). Strategijo iskanja v širino uporablja algoritem *iskanja po labirintu* (angl. *maze router*) oziroma *Leejev algo-*

ritem [17], strategijo iskanja v globino pa uporablja algoritem *iskanja po liniji* (angl. *line probe search*) [100, 101]. Primer uporabe iskanja po načelu najprej najboljši z uporabo hevristike je *algoritem A** [18]. Naštete strategije se uporabljajo tako za globalno kot tudi za podrobno povezovanje.

6.4.1 Uporaba iskanja po labirintu

Iskanje po labirintu je eden izmed prvih in najpogosteje uporabljanih algoritmov za iskanje povezave med dvema točkama. Površina je predstavljena z mrežo, na kateri se nahajata izvorna in ponorna točka. Nekatera področja v mreži so že zasedena, kot je prikazano v primeru na sliki 6.2(a). Povezava se lahko namesti na nezasedena področja,



Slika 6.2 Mrežo sestavljajo osenčena zasedena področja in ostala prosta področja, na katere se lahko namesti povezava (a). Področje z izvorno točko je označeno z I, ponorno področje pa s P. Slika (b) prikazuje označevanje prostih področij, ki poteka od izvora proti ponoru. Ko je dosežen ponor, se področja za namestitev povezave določi z vračanjem od ponora do izvora (c).

tako da zaseda izbrana sosednja področja od izvorne do ponorne točke.

Psevdokoda iskanja po labirintu je zapisana v algoritmu 2. Algoritem je sestavljen iz dveh faz. V prvi fazi se označujejo prosta področja glede na njihovo oddaljenost od izvora. Označevanje, ki poteka kot širjenje vala, je prikazano na sliki 6.2(b). Najprej dobi izvorna točka oznako 0. V prvi iteraciji se vsem področjem v njeni von Neumannovi sosednosti z manhattansko razdaljo 1 pripiše za 1 večja oznaka. V naslednji iteraciji se poiščejo vsi še neoznačeni sosedi področij, ki so bila označena v prejšnji iteraciji. Tem neoznačenim sosedom se pripiše za 1 večja oznaka, kot jo imajo v prejšnji iteraciji označena področja. Postopek se iterativno nadaljuje, dokler ni označeno področje s ponorno točko. Oznaka kateregakoli področja ustreza manhattanski razdalji med tem področjem in izvornim področjem, pri čemer je upoštevano izogibanje zasedenim področjem.

V drugi fazi se z vračanjem določijo področja, na katerih bo nameščena povezava.

Algoritem 2 Iskanje po labirintu.

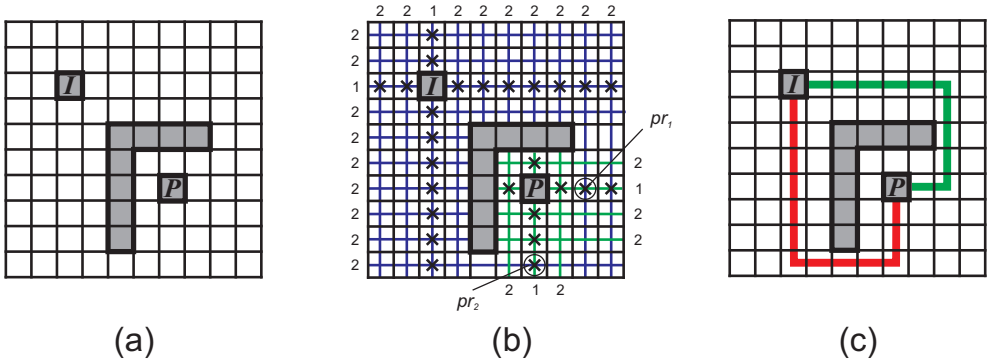
```
procedure ISKANJE_PO_LABIRINTU
// označevanje področij
// označi izvorno področje z oznako 0
    označi(izvor, 0)
// množica S1 na začetku vsebuje sosede izvornega področja
    S1 = sosedi(izvor); oznaka = 1
    for vsako področje s ∈ S1 do
        označi(s, oznaka)
    end for
     // obravnava vsa področja v množici S1
    while s ∈ S1, s != ponor do
        oznaka = oznaka + 1
     // S2 je množica neoznačenih sosedov področij v S1
        for vsako področje t ∈ S2 do
            označi(t, oznaka)
        end for
     // v naslednji iteraciji se bodo obravnavala novo označena področja v S2
        S1 = S2
    end while

// določanje področij za namestitev povezave
// postopek se začne pri ponoru
    p = ponor; oznaka = oznaka(ponor)
// vračanje do izvornega področja
    while p != izvor do
        oznaka = oznaka - 1
     // izberi soseda z oznako, ki je za 1 manjša od oznake obravnavanega področja p
        p = sosed(p, oznaka)
        določi_za_povezavo(p)
    end while
end procedure
```

Postopek vračanja se začne pri ponorni točki, nato pa se izbere področje, ki ima za 1 manjšo oznako. Tako se iterativno izbirajo področja z vse manjšimi oznakami, dokler ni izbrano izvorno področje. Povezava se namesti v izbrana področja, kot je prikazano na sliki 6.2(c).

Iskanje po labirintu uporablja princip iskanja v širino v prostoru stanj, ki zagotavlja, da je rešitev mogoče najti. Vsaka tako najdena rešitev ima minimalno dolžino. Slabost algoritma iskanja po labirintu je njegova velika časovna zahtevnost, ki je enaka $O(W \cdot H)$, pri čemer je W širina in H višina mreže. Obe dimenziji sta merjeni v številu področij.

6.4.2 Uporaba iskanja po liniji

Tako kot pri iskanju po labirintu, je tudi pri iskanju po liniji površina predstavljena z mrežo, na kateri se nahajata izvorna in ponorna točka, nekatera področja pa so že zasedena. Primer mreže je prikazan na sliki 6.3(a). Algoritem iskanja po labirintu obravnava



Slika 6.3 Mrežo sestavljajo osenčena zasedena področja in ostala prosta področja, na katere se lahko namesti povezava (a). Področje z izvorno točko je označeno z I, ponorno področje pa s P. Na sliki (b) so prikazane linije, ki potekajo skozi področja. Linije, ki pripadajo množici I_L, so obarvane z modro, linije iz množice P_L pa z zeleno barvo. S križci (×) so označena področja na linijah, položenih v prvem koraku. Skozi označena področja se v drugem koraku položijo linije, ki so pravokotne na v prvem koraku položene linije. Ob vsaki liniji je na robu mreže zapisana zaporedna številka koraka, v katerem je bila linija položena. Vsako presečišče pr_1 in pr_2 določa presečišče med eno linijo iz množice I_L in eno linijo iz P_L. Slika (c) prikazuje dva možna poteka povezave med I in P. Zelena povezava poteka skozi presečišče pr_1, rdeča povezava pa skozi pr_2. Obe povezavi vsebujeta dva kota, razlikujeta pa se v dolžini, saj je zelena povezava krajša od rdeče.

vsako področje v mreži posebej, pri iskanju po liniji pa se išče povezava med dvema točkama z uporabo linij, ki potekajo skozi večje število področij naenkrat.

Psevdokoda iskanja po liniji je navedena v algoritmih 3 in 4. Podobno kot iskanje po labirintu je tudi algoritem iskanja po liniji sestavljen iz dveh faz. V prvi fazi se

Algoritem 3 Iskanje po liniji 1.

procedure ISKANJE_PO_LINIJI

// **polaganje linij**

// množici I'_L in P'_L vsebujeta linije, položene v trenutnem koraku

// u_I je vodoravna linija skozi I, v_I je navpična linija skozi I

// u_P je vodoravna linija skozi P, v_P je navpična linija skozi P

$I_L = I'_L = \{u_I, v_I\}$; $P_L = P'_L = \{u_P, v_P\}$

// polaga linije, dokler ne najde presečišča

while ne obstaja presečišče med linijo v I_L in linijo v P_L **do**

// polaganje linij, ki so pravokotne na linije v množici I'_L

$i = 0$

for vsako linijo $a_i \in I'_L$ **do**

// množica A_i vsebuje linije, ki so pravokotne na a_i

$A_i = \emptyset$

for vsako področje p na liniji a_i **do**

// v A_i doda linijo, ki je pravokotna na a_i in poteka skozi področje p

$A_i = A_i \bigcup$ pravokotna_linija(a_i, p)

end for

$i = i + 1$

end for

// množica I'_L vsebuje vse linije, ki so bile položene v trenutnem koraku

$I'_L = \bigcup_{k=0}^{k=i-1} A_k$; $I_L = I_L \bigcup I'_L$

// polaganje linij, ki so pravokotne na linije v množici P'_L

$j = 0$

for vsako linijo $b_j \in P'_L$ **do**

$B_j = \emptyset$

for vsako področje r na liniji b_j **do**

$B_j = B_j \bigcup$ pravokotna_linija(b_j, r)

end for

$j = j + 1$

end for

$P'_L = \bigcup_{k=0}^{k=j-1} B_k$; $P_L = P_L \bigcup P'_L$

end while

Algoritem 4 Iskanje po liniji 2.

```
// določanje področij za namestitev povezave
// postopek se začne pri presečišču
   i = 0
// E je seznam povezav, razvrščenih glede na dolžino od najkrajše do najdaljše
   E = ∅
   for vsako presečišče p_i med linijo v I_L in linijo v P_L do
      s = t = p_i; e_i = ∅
    // vračanje po linijah do izvornega področja
      while s != izvor do
        // doda s v povezavo e_i
         dodaj(s, e_i)
        // izbiranje področij poteka po linijah v I_L v smeri proti izvoru
         s = sosed(s, I_L)
      end while
    // vračanje po linijah do ponornega področja
      while t != ponor do
        // doda t v povezavo e_i
         dodaj(t, e_i)
        // izbiranje področij poteka po linijah v P_L v smeri proti ponoru
         t = sosed(t, P_L)
      end while
      i = i + 1
    // vstavi e_i na ustrezno mesto v E
      vstavi(e_i, E)
   end for
// namesti se najkrajša povezava
return prvi_element(E)
end procedure
```

na področja iterativno polagajo ravne linije, kot je prikazano na sliki 6.3(b). V prvem koraku iteracije se položita vodoravna in navpična linija, ki potekata skozi izvorno točko. Vsaka linija poteka do prvega zasedenega področja oziroma do roba mreže. Ti dve liniji na začetku tvorita množico I_L. Prav tako se v prvem koraku iteracije položita tudi vodoravna in navpična linija, ki potekata skozi ponorno točko. Tudi ti dve liniji lahko potekata le skozi prosta področja. Na začetku tvorita množico P_L. V naslednjem koraku iteracije se označijo vsa področja, skozi katera potekajo linije, ki so bile položene v prejšnjem koraku. Nato se položijo linije, ki potekajo skozi označena področja in so pravokotne na linije, položene v prejšnjem koraku. Če je v trenutnem koraku položena linija l_i pravokotna na linijo $l_j \in I_L$, se l_i doda v množico I_L, sicer pa se doda v množico P_L. Postopek se iterativno nadaljuje, dokler ne pride do presečišča med neko linijo v množici I_L in eno izmed linij v P_L. Iskanje se ustavi, ko je določeno presečišče, kot je prikazano na sliki 6.3(b).

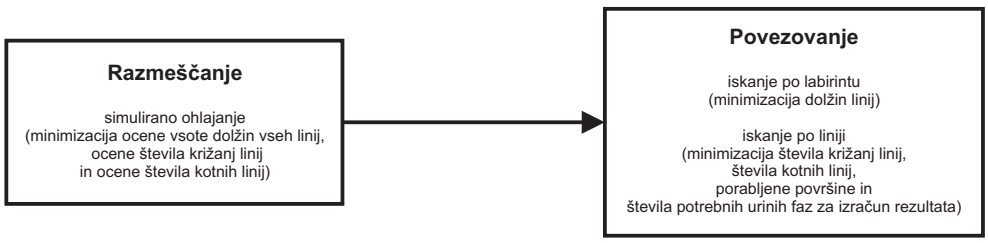
V drugi fazi algoritma se določijo področja, na katera bo nameščena povezava med izvorom in ponorom. Postopek se začne pri presečišču in se nadaljuje po položenih linijah tako v smeri proti izvoru, kot tudi v smeri proti ponoru. Ko sta dosežena tako izvor kot ponor, se povezava namesti v izbrana področja, kot je prikazano na sliki 6.3(c). Obstaja lahko več presečišč, ki določajo različne možne lege povezav. Povezave, ki potekajo skozi različna presečišča, imajo lahko različne dolžine. Zato se izbere presečišče, pri katerem ima nameščena povezava manjšo dolžino od ostalih možnosti.

Z iskanjem po liniji je vedno mogoče najti rešitev, če ta obstaja. Vsaka rešitev določa povezavo z minimalnim številom kotov, ni pa nujno, da ima tudi minimalno dolžino. Ker algoritem iskanja po liniji ne obravnava vsakega področja posebej, je prostorsko in časovno manj zahteven od algoritma iskanja po labirintu.

7 Postopki snovanja fizične razmestitve QCA strukture

7.1 Uvod

Shema delovanja algoritma razmeščanja in povezovanja je prikazana na sliki 7.1. Najprej



Slika 7.1 Shema delovanja algoritma razmeščanja in povezovanja.

se za vsak logični primitiv določi, v katere urine cone je lahko nameščen. Nato se izvede razmeščanje logičnih primitivov s tehniko simuliranega ohlajanja, pri čemer se minimizirajo ocena vsote dolžin vseh linij, ocena števila križanj linij in ocena števila kotnih linij. Razmeščene primitive se nato v fazi povezovanja poveže z linijami z uporabo kombinacije tehnik iskanja po labirintu in iskanja po liniji. Pri tem se minimizirajo dolžine linij,

število križanj, število kotnih linij, poraba površine in število potrebnih urinih faz za izračun rezultata.

Metode snovanja struktur kvantnih celičnih avtomatov temeljijo na metodah snovanja CMOS vezij, ki so se izkazale za uporabne. Vendar je potrebne pri snovanju struktur QCA upoštevati njihove specifične lastnosti, ki se razlikujejo od CMOS tehnologije. Za razliko od CMOS so QCA v celoti sestavljeni iz enega samega tipa elementov, to je QCA celic. Križanje povezav je v CMOS možno na številnih nivojih, v QCA pa je trenutno opisano križanje le na dveh nivojih. Posebnost QCA je tudi razdelitev strukture na urine cone.

Ročno razvite razmestitve niso zasnovane s formalnim postopkom, zato jih ni možno posplošiti za uporabo pri snovanju poljubne strukture. Metode za avtomatsko snovanje poskrbijo tudi za dosledno upoštevanje načrtovalskih pravil. Pri neformalnem ročnem snovanju kompleksnih struktur lahko pride do neupoštevanja pravil. Tako zasnovana struktura je lahko tehnološko neizvedljiva in je torej neuporabna. Snovanje z avtomatiziranimi metodami poteka hitreje kot ročno snovanje, poleg tega pa lahko avtomatske metode izvedejo boljšo optimizacijo.

7.2 Razdelitev logičnih primitivov v urine cone

Za vsak logični primitiv lp se določi seznam urinih con, v katere je lp lahko nameščen. Določanje con za posamezen primitiv je predstavljeno s psevdokodo v algoritmu 5. Geometrija strukture je določena s conami, ki si zaporedno sledijo od cone UC_1 z vhodnimi priključki strukture do cone UC_n z izhodnimi priključki strukture. Cone so razvrščene v vrsti od UC_1 na levi strani geometrije do UC_n na desni strani. Položaji primitivov znotraj celotne geometrije so soodvisni. Če poteka povezava od izhodnega priključka na primitivu lp_i do vhodnega priključka na primitivu lp_j, se mora lp_i nahajati levo od lp_j. Če je lp_i nameščen v cono UC_u in lp_j v cono UC_v, mora torej veljati $u < v$.

Logični primitivi se nameščajo v cone s primitivi UC_{L1}, UC_{L2}, ..., UC_{Lp}. Med temi conami ležijo vmesne cone, ki so namenjene za namestitev povezav, zato je njihovo število znano šele po zaključku faze povezovanja. Za definiranje soodvisnosti položajev logičnih primitivov se sestavi usmerjeni graf logične sheme $G(V, E)$. Vozlišča v grafu ustrezajo primitivom v logični shemi strukture. Usmerjene povezave v grafu predstavljajo povezave med primitivi v logični shemi strukture. Usmerjeni graf logične sheme se topološko uredi,

Algoritem 5 Razdelitev logičnih primitivov v urine cone.

procedure RAZDELITEV_LOGIČNIH_PRIMITIVOV($G(V, E)$)
vhodni podatek: $G(V, E)$ je usmerjeni graf logične sheme
// topološko uredi $G(V, E)$
topološko_urejanje($G(V, E)$)
// Število con z logičnimi primitivi določa dolžina kritične poti v $G(V, E)$
p = dolžina(kritična_pot($G(V, E)$))
// vozlišče lp' predstavlja primitiv lp
for vsak $lp' \in V$ **do**
// najmanjši indeks cone z lp je enak maksimalni dolžini poti med vozliščem lp'
// in katerimkoli začetnim vozliščem v grafu
// poti med lp' in začetnimi vozlišči so v seznamu $Poti_1$ urejene po padajoči dolžini
min = prvi_element($Poti_1$)
// največji indeks cone z lp je enak maksimalni dolžini poti med vozliščem lp'
// in katerimkoli končnim vozliščem v grafu
// poti med lp' in končnimi vozlišči so v seznamu $Poti_2$ urejene po padajoči dolžini
max = prvi_element($Poti_2$)
// lp se lahko namesti v katerokoli cono med UC_{min} in UC_{max} vključno s tema dvema
// conama
določi_možne_cone(lp, min, max)
end for
end procedure

kar določa soodvisnosti položajev primitivov. Naj vozlišče lp'_i v grafu predstavlja primitiv lp_i, vozlišče lp'_j pa naj predstavlja primitiv lp_j. Vzemimo, da se lp'_i nahaja pred lp'_j v topološki urejenosti grafa ($lp'_i < lp'_j$). Če je lp_i nameščen v coni UC_u in lp_j v coni UC_v, mora veljati $u < v$.

Število urinih con z logičnimi primitivi p je določeno z dolžino kritične poti v usmerjenem grafu logične sheme. Vsak primitiv lp_i, predstavljen z vozliščem lp'_i na kritični poti v grafu, mora biti nameščen v točno določeno cono. Če se lp'_i nahaja na k-tem mestu na kritični poti, se lp_i namesti v cono UC_{Lk}. Če vozlišče lp'_j ne leži na kritični poti, obstaja več možnih con za namestitev ustreznega primitiva lp_j, pri tem pa je vedno potrebno upoštevati soodvisnosti, določene s topološko urejenostjo grafa.

7.3 Uporaba simuliranega ohlajanja za razmeščanje v QCA tehnologiji

Simulirano ohlajanje se uspešno uporablja pri fizičnem snovanju CMOS integriranih vezij, zato smo ga uporabili tudi za fizično snovanje kvantnih celičnih avtomatov. Uporaba simuliranega ohlajanja za razmeščanje logičnih primitivov v QCA je podana s psevdokodo v algoritmu 6. Ključni deli algoritma so:

- **Izbira logičnega primitiva ali vhodno/izhodnega priključka:** Naključno z enakomerno porazdeljeno verjetnostjo se izbere logični primitiv ali vhodno/izhodni priključek. Za izbrani gradnik se nato določi poteza, s katero se spremeni njegov položaj. Pri tem se upošteva razdelitev logičnih primitivov v urine cone glede na topološko urejenost grafa logične sheme. Vhodni priključki logične strukture se morajo nahajati na levi strani prve urine cone, izhodni priključki pa na desni strani zadnje urine cone.

- **Poteze za prehod v novo stanje:** Stanja so določena s položaji gradnikov. Prehod v novo stanje je določen s spremembo položaja gradnika. Tako se lahko logičnim primitivom in vhodno/izhodnim priključkom določi nov položaj, pri čemer ne sme priti do prekrivanja z ostalimi gradniki. Ker se vsi vhodni priključki strukture nahajajo v isti coni in imajo enake velikosti, se lahko enostavno zamenjata položaja dveh vhodnih priključkov. Podobno se lahko zamenjata položaja dveh izhodnih priključkov strukture. Logični primitivi nimajo vsi enakih oblik, poleg tega pa so njihovi položaji medsebojno soodvisni glede na topološko urejenost usmerjenega grafa logične sheme. Zato njihovih položajev nismo zamenjevali. Prehod v novo stanje predstavlja tudi podaljšanje navpičnega dela kotnih linij v nadgrajenih majoritetnih vratih, s čimer se spremenita položaja njihovega zgornjega (x_1) in spodnjega (x_3) vhodnega priključka. Ker je majoritetna funkcija komutativna, se lahko tudi poljubno permutirajo položaji njihovih vhodnih priključkov.

  Ker se vsi vhodni priključki nahajajo na levi strani prve urine cone, se pri določanju novega položaja vhodnega priključka spremeni le njegova y koordinata. Prav tako se spremeni le y koordinata pri določanju novega položaja izhodnega priključka, saj vsi izhodni priključki ležijo na desni strani zadnje urine cone. Vsi logični primitivi imajo enako širino kot urina cona, zato se tudi njim spremeni le y koordinata pri

Algoritem 6 Razmeščanje primitivov v QCA z uporabo simuliranega ohlajanja.

procedure RAZMEŠČANJE
  $T = T_0$; $i = 0$
// naključno določi začetno razmestitev in njeno ceno
  s_i = naključna_razmestitev(); c = cena(s_i)
  $sbest = s_i$; $cbest = c$
  while $T > \epsilon$ **do**
    $i = i + 1$
    // naključno izberi logični primitiv lp iz množice primitivov P
    lp = naključno_izberi(P)
    // $r_1 \in [0,1)$ in $r_2 \in [0,1)$ sta naključno izbrani števili
    if $r_1 < p_{nov\_polozaj}$ **then**
      // izbira novega položaja; (x, y) je naključno izbran položaj
      namesti(lp, (x, y))
    else if $r_1 < p_{nov\_polozaj} + p_{zamenjava}$ **then**
      // zamenjava položaja
      lp' = naključno_izberi(P)
      zamenjaj_položaja(lp, lp')
    else
      // permutacija vhodnih priključkov, če lp spada med majoritetna vrata
      permutiraj_vhode(lp)
    end if
    c = cena(s_i)
    if $c < cbest$ **then**
      $sbest = s_i$; $cbest = c$
    end if
    if $p(s_i|s_{i-1}) < r_2$ **then**
      // prehod v stanje s_i se ne izvede
      $s_i = s_{i-1}$; c = cena(s_i)
    end if
    T = ohlajanje(T, i)
  end while
end procedure

premiku znotraj iste cone. Do spremembe x koordinate logičnega primitiva pride le v primeru, ko se primitiv prestavi v drugo urino cono. Naj bo y koordinata spodnjega levega področja v coni enaka $y_{spodnja}$ in y koordinata zgornjega levega področja enaka $y_{zgornja}$. Ker mora vsak gradnik ležati znotraj urine cone, mora za njegovo y koordinato $y_{gradnik}$ veljati $y_{spodnja} \leq y_{gradnik} \leq y_{zgornja}$.

V primerih, ko se spreminja le y koordinata gradnika, se slednja lahko spremeni za največ d_i. Na začetku je d_i enak višini urine cone $H(UC)$, tako da je lahko razdalja med novim in starim položajem gradnika velika. Med izvajanjem algoritma se d_i manjša, tako da so proti koncu možni vse manjši premiki. S tem postaja optimizacija vse bolj lokalna. Proti koncu delovanja algoritma je razmestitev vse bolj optimizirana, zato veliki premiki običajno pripeljejo do razmestitve z višjo ceno. Parameter d_i se v i-ti iteraciji izračuna z izrazom

$$d_i = \begin{cases} H(UC) \cdot (1 - \frac{i}{N_{iteracij}}) & \text{če } H(UC) \cdot (1 - \frac{i}{N_{iteracij}}) \geq 5 \\ 5 & \text{sicer,} \end{cases} \tag{7.1}$$

kjer je $N_{iteracij}$ število vseh iteracij. Minimalna vrednost parametra d_i je enaka 5.

- **Verjetnost izbire poteze:** Če so v prvi fazi izbrana majoritetna vrata, se naključno določi njihov nov položaj z verjetnostjo $p_{nov\_polozaj\_mv} = 1/3$. Verjetnost podaljšanja navpičnega dela kotnih linij v nadgrajenih majoritetnih vratih za naključno izbrano število med 0 in 3, je enaka $p_{podaljsanje} = 1/3$. Tudi verjetnost permutacije vhodov nadgrajenih majoritetnih vrat je enaka $p_{permutacija} = 1/3$. Če je v prvi fazi izbran logični primitiv negator, je edina možnost za prehod v novo stanje določitev novega položaja negatorja. Če je izbran vhodni priključek strukture, se mu določi nov naključno izbran položaj z verjetnostjo $p_{nov\_polozaj\_vhod} = 1/2$. Z verjetnostjo $p_{zamenjava\_vhod} = 1/2$ se naključno izbere enega od preostalih vhodnih priključkov, nato pa se zamenjata njuna položaja. Podobno velja za izbran izhodni priključek strukture. Z verjetnostjo $p_{nov\_polozaj\_izhod} = 1/2$ se mu naključno določi nov položaj, z verjetnostjo $p_{zamenjava\_izhod} = 1/2$ pa zamenja položaj z enim izmed preostalih izhodnih priključkov.

- **Izračun cene trenutne razmestitve:** Izračun cene trenutne razmestitve mora biti izveden hitro, ker se izračuna v vsaki iteraciji. Zato smo za vrednosti kriterijev izračunali ocene, dejanske vrednosti pa so dokončno določene šele po končani fazi

povezovanja. Cena nove razmestitve se izračuna z uteženo vsoto ocene dolžine linij, ocene števila križanj linij in ocene števila kotnih linij po formuli

$$\text{cena}(s) = k_1 \cdot \sum_i N'_{dolzina_i} + k_2 \cdot N'_{krizanje} + k_3 \cdot N'_{kot}. \tag{7.2}$$

cena(s) je cena trenutne razmestitve v stanju s, $\sum_i N'_{dolzina_i}$ ocena dolžine linij, $N'_{krizanje}$ ocena števila križanj linij in N'_{kot} ocena števila kotnih linij. Koeficienti k_1, k_2 in k_3 določajo uteži. Vsota dolžin linij v strukturi je veliko večja kot število kotnih linij ali število križanj linij, zato mora biti koeficient k_1 ustrezno manjši od k_2 in k_3. Za vrednosti koeficientov smo tako določili $k_1 = 1$, $k_2 = 5$ in $k_3 = 5$. Za izračun ocen smo najprej določili trenutne lege vseh povezav. Vsaka povezava je sestavljena iz linij, tako da vsaka linija leži v svoji urini coni. Lega vsake linije je določena tako, da je linija znotraj urine cone čim krajša. Takšna namestitev linij je optimalna glede na dolžino le lokalno v urini coni, lahko pa obstaja krajša celotna povezava, ki poteka skozi večje število con. Iskanje najkrajše celotne povezave je računsko potratno, naš postopek določanja najkrajše linije v urini coni pa se lahko izvede hitro. Ocena skupne dolžine linij $\sum_i N'_{dolzina_i}$ je vsota dolžin vseh tako nameščenih linij. Če se QCA celica v liniji l_i na levem robu urine cone ne nahaja na isti višini kot celica v l_i na desnem robu iste urine cone, to pomeni, da je l_i kotna linija. Število vseh trenutno nameščenih kotnih linij je enako N'_{kot}. Za izračun ocene števila križanj linij se najprej določi interval $[y_1, y_2]$, kjer je y_1 y koordinata celice v liniji l_i na levem robu urine cone in y_2 y koordinata celice v l_i na desnem robu iste cone. Do križanja pride takrat, kadar linija l_j poteka skozi interval $[y_1, y_2]$. Naj bo z_1 y koordinata celice v l_j na levem robu cone in z_2 y koordinata celice v l_j na desnem robu cone. Linija l_j poteka skozi interval $[y_1, y_2]$ takrat, ko velja $z_1 > y_1 \wedge z_2 < y_2$ ali $z_1 < y_1 \wedge z_2 > y_2$. Število vseh križanj linij je enako $N'_{krizanje}$.

Stanje z najnižjo ceno, ki je enaka *cbest*, hrani spremenljivka *sbest*. Če ima novo stanje s nižjo ceno od *cbest*, se s zapiše v spremenljivko *sbest*, cena(s) pa se zapiše v spremenljivko *cbest*.

- **Izbira funkcije ohlajanja:** Funkcija ohlajanja predpisuje spreminjanje temperature glede na iteracijo. Število vseh iteracij $N_{iteracij}$ mora biti dovolj veliko, da se preišče čim večji del prostora stanj. V nasprotnem primeru bi se iskanje prehitro ustavilo v lokalnem minimumu, ki se nahaja blizu začetne razmestitve. Za

število vseh iteracij smo določili $N_{iteracij}$=1 000 000. Ista vrednost temperature se uporabi v konstantnem številu $N_{iter/temp}$ zaporednih iteracij. Tako se pri isti vrednosti temperature preišče del prostora stanj. Število $N_{iter/temp}$ ne sme biti preveliko, sicer bi se med celotnim izvajanjem algoritma izračunalo majhno število različnih vrednosti temperature. Izračun nove vrednosti temperature se izvede po vsakih $N_{iter/temp}$=500 zaporednih iteracijah. Ista vrednost temperature se torej uporabi v 500 zaporednih iteracijah, nato pa se izračuna nova vrednost. Ta postopek se med delovanjem ponavlja, tako da je v celotnem izvajanju algoritma število izračunov novih vrednosti temperature enako $N_{izracunov} = \frac{N_{iteracij}}{N_{iter/temp}}$=2000.

Izbrali smo pogosto uporabljano eksponentno ohlajanje [12]. Temperatura T_i v i-tem izračunu nove vrednosti se izračuna z izrazom

$$T_i = k_4 T_{i-1}. \tag{7.3}$$

Temperatura T_i je tako enaka $T_0 k_4^i$. Začetna temperatura T_0 mora biti dovolj visoka, končna temperatura T_{koncna} pa dovolj nizka. Za vrednost začetne temperature smo določili $T_0 = 10^6$ in za vrednost končne temperature $T_{koncna} = 0,1$. Koeficient k_4 je število med 0 in 1, torej $0 < k_4 < 1$. Med izvajanjem algoritma se mora temperatura znižati od T_0 do T_{koncna} v $N_{izracunov}$ izračunih nove vrednosti. Končna temperatura je enaka $T_{koncna} = T_0 k_4^{N_{izracunov}}$, zato se koeficient k_4 izračuna z izrazom

$$k_4 = \sqrt[N_{izracunov}]{T_{koncna}/T_0} \approx 0,9919. \tag{7.4}$$

- **Verjetnost prehoda v novo stanje:** Funkcija za izračun verjetnosti prehoda $p(s_i|s_{i-1})$ v stanje s_i iz stanja s_{i-1} je

$$p(s_i|s_{i-1}) = \begin{cases} 1 & \text{če cena}(s_i) \leq \text{cena}(s_{i-1}) \\ k_5 \cdot e^{-k_6(\text{cena}(s_i) - \text{cena}(s_{i-1})/T_i} & \text{sicer.} \end{cases} \tag{7.5}$$

Kadar ima stanje s_i višjo ceno od stanja s_{i-1}, je minimalna razlika med njunima cenama enaka

$$\text{cena}(s_i) - \text{cena}(s_{i-1}) = 1. \tag{7.6}$$

Parametra k_5 in k_6 smo določili tako, da je na začetku delovanja algoritma verjetnost prehoda v stanje z višjo ceno visoka, proti koncu pa se približa 0. Pri razliki

med cenama, izračunani z enačbo (7.6), smo za verjetnost prehoda v stanje z višjo ceno v prvi iteraciji določili vrednost 0,95, po 90% izvedenih iteracij (v iteraciji z indeksom 900 000) pa se verjetnost zmanjša na 0,1. Koeficienta k_5 in k_6 se tako izračunata z izrazoma

$$k_6 = \ln(\frac{0,95}{0,1}) \cdot T_0 \cdot k_4^{0,9 \cdot N_{izracunov}} \approx 1,1283 \quad \text{in} \tag{7.7}$$

$$k_5 = 0,95 \cdot e^{k_6/T_0} \approx 0,95. \tag{7.8}$$

Verjetnost prehoda v stanje z višjo ceno je na začetku izvajanja algoritma visoka, nato pa se približuje številu 0. Tako se na začetku algoritem izogiba lokalnim minimumom. Proti koncu izvajanja, ko je cena najboljšega stanja že precej nizka, so vse bolj možni le še prehodi v stanja z nižjo ceno.

V zadnjem procentu vseh iteracij algoritem izvede požrešno lokalno optimizacijo. V primeru, ko je vseh iteracij 1 000 000, zadnji procent iteracij predstavlja 10 000 iteracij med iteracijo z zaporednim indeksom 990 000 in zadnjo iteracijo z indeksom 1 000 000. V postopku lokalne požrešne optimizacije algoritem najprej izbere najboljšo do takrat najdeno razmestitev. Nato poskuša slednjo optimizirati s požrešnim algoritmom. Ker pri požrešni optimizaciji prehod v stanje z višjo ceno ni mogoč, algoritem takrat poišče lokalni minimum s_{min}. Slednji se nahaja v bližini najboljše razmestitve, ki jo je algoritem našel v prvih 99 procentih iteracij. Zato je cena lokalnega minimuma s_{min} običajno nižja od cene lokalnega minimuma t_{min}, ki se nahaja v bližini začetnega stanja z visoko ceno. Stanje t_{min} bi bila rešitev algoritma, ki bi v vseh iteracijah izvajal požrešno iskanje. Zato je rezultat s_{min}, ki ga najde algoritem simuliranega ohlajanja, cenejši od rezultata požrešnega algoritma t_{min}.

7.4 Povezovanje logičnih primitivov

V fazi povezovanja logičnih primitivov v QCA se vsaka povezava v logični strukturi realizira z linijo v fizični razmestitvi strukture. Vsaki liniji se določi natančen položaj v fizični razmestitvi. To se izvede tako, da se za vsako QCA celico v liniji določi področje v mreži, v katerega bo celica nameščena. Pri večnivojskem križanju je potrebno za vsako celico opredeliti tudi plast, na kateri se celica nahaja.

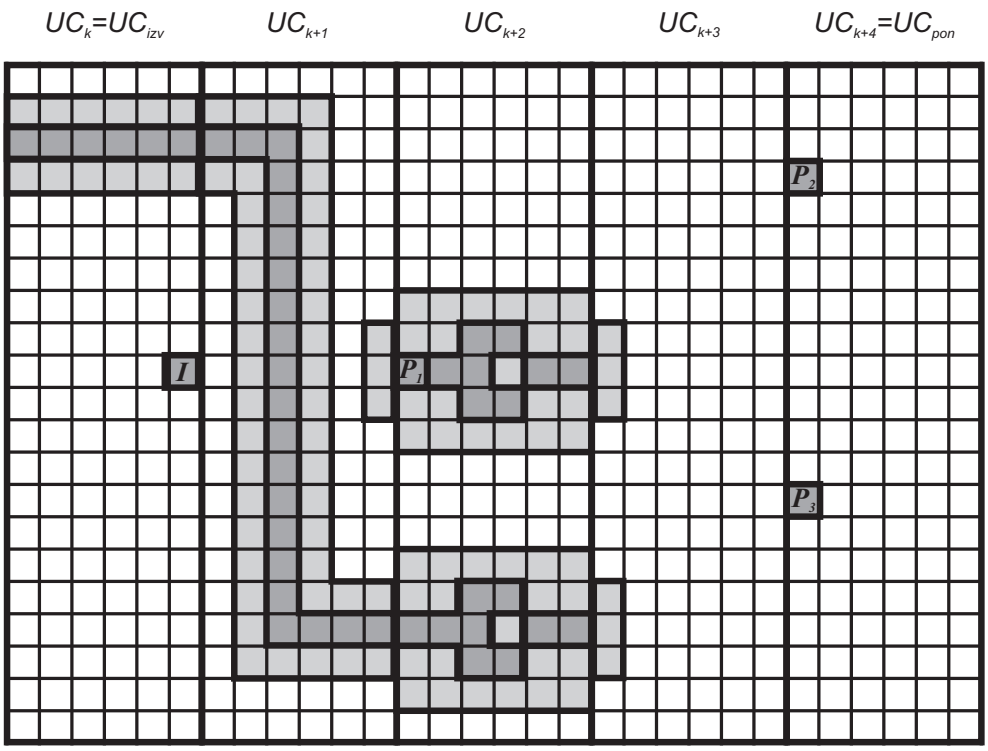
Za povezovanje v QCA potrebujemo algoritem, ki določi področja v mreži, v katere bodo nameščene celice v liniji. Algoritem mora omogočati izogibanje predhodno name-

ščenim QCA celicam. Zato smo za povezovanje izbrali algoritma iskanja po labirintu ter iskanja po liniji in ju priredili za uporabo v QCA.

Skozi urino cono z logičnimi primitivi lahko poteka le vodoravna linija, ki se ne prekriva z nobenim primitivom. Vse kotne in razvejitvene linije so nameščene v vmesnih urinih conah, zato da je preprečeno njihovo prekrivanje s primitivi. Poleg kotnih in razvejitvenih linij se znotraj vmesnih urinih con nahajajo tudi preostale potrebne vodoravne linije. Do križanja lahko pride le med vodoravno linijo in navpičnim odsekom kotne linije, zato se križanja linij nahajajo le v vmesnih conah. S tem je onemogočena situacija, pri kateri bi se linija, priključena na logični primitiv p_i v urini coni UC_{Lj}, v isti urini coni križala z neko drugo linijo. To bi privedlo do napačnega delovanja, saj mora biti zaporedje procesiranja v logičnem primitivu in nato v križanju linij razdeljeno na ločeni urini coni.

Posamezna povezava povezuje en izhodni priključek z enim ali več vhodnih priključkov. Izhodni priključek je lahko QCA celica, ki predstavlja vhodni priključek strukture, ali pa celica, ki se nahaja na desni strani logičnega primitiva. Vsak vhodni priključek je ali celica, ki predstavlja izhodni priključek strukture, ali ena od celic na levi strani majoritetnih vrat oziroma negatorja. Izvorna točka povezave se nahaja na desnem robu urine cone, v kateri leži izhodni priključek. Če slednji ne leži na desnem robu cone, se poveže z izvorno točko tako, da se med njima namesti vodoravno linijo. Vsaka ponorna točka povezave leži na levem robu cone, v kateri leži pripadajoč vhodni priključek. Če ponorna točka in vhodni priključek prostorsko ne sovpadata, se povežeta z vodoravno linijo. Na sliki 7.2 je prikazana mreža, na kateri se nahaja eno področje z izvorno točko (I) in tri področja s ponorno točko (P_1, P_2 in P_3). Nekatera področja so zasedena z že nameščenimi QCA celicami. Za zasedena se označijo tudi nekatera druga področja, kot to določajo načrtovalska pravila. Tako so zasedena tudi vsa področja, ki so sosednja področju s QCA celico, kakor določa pravilo o minimalni razdalji med logičnimi primitivi.

Vrstni red povezav se najprej naključno premeša, s čimer je možno dobiti bolj ugoden vrstni red nameščanja od začetnega. Nato se povezave na površino nameščajo zaporedno od e_1 do e_{N_3}, kjer je N_3 število vseh povezav. Področja, ki jih zasedajo celice v že nameščenih povezavah e_1, e_2, ..., e_i, se označijo za zasedena, kot v primeru na sliki 7.2. Na že zasedena področja se ne morejo namestiti celice v katerikoli povezavi e_j, $i < j$, ki se bo namestila po e_i. Vsaka povezava je sestavljena iz zaporedno povezanih linij, ki potekajo od izvorne točke do ponorov, tako da je vsaka linija nameščena v svoji urini

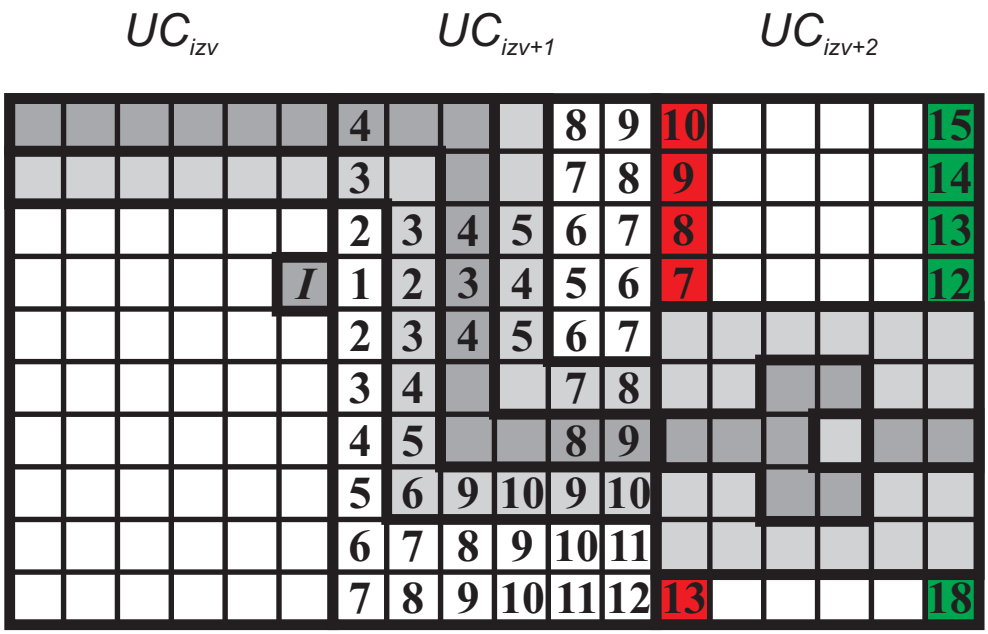


Slika 7.2 Primer mreže kvantnega celičnega avtomata. Prikazanih je pet urinih con (UC_k, UC_{k+1}, UC_{k+2}, UC_{k+3} in UC_{k+4}). Področje z izvorno točko je označeno z I in se nahaja v coni $UC_k = UC_{izv}$. Ponorna točka P_1 leži v UC_{k+2}, ponora P_2 in P_3 pa v $UC_{k+4} = UC_{pon}$. Zasedena področja so osenčena. S temnejšim odtenkom so obarvana področja s predhodno nameščenimi QCA celicami. Slednje se nahajajo v vodoravni liniji v coni UC_k, v kotni liniji v UC_{k+1} in v dveh negatorjih v UC_{k+2}. Področja, ki so zasedena zaradi načrtovalskih pravil, so obarvana svetlo sivo.

coni. Vodoravne linije lahko ležijo v katerikoli coni med cono UC_{izv} z izvorno točko in najbolj desno cono UC_{pon} s ponorno točko. Kotne in razvejitvene linije se morajo namestiti v vmesne cone med UC_{izv} in UC_{pon}.

Povezava e_i z m ponori povezuje izvorno točko I s ponornimi točkami P_1, P_2, ..., P_m. Ponori so razvrščeni tako, da se ponor P_u nahaja v isti coni kot P_v, ali pa desno od te cone, če velja $u > v$. Naj P_u leži v coni $UC_{u'}$ in P_v v coni $UC_{v'}$. Tedaj velja $(u > v) \Rightarrow (u' \geq v')$.

Najprej se v stopnji globalnega povezovanja poišče prostor za namestitev povezave e_i. Globalno povezovanje se izvede z uporabo iskanja po labirintu. Označevanje se prične pri izvorni točki in se nadaljuje skozi cone med UC_{izv} in UC_{pon}, dokler niso doseženi vsi ponori. Označevanje v vmesni coni je prikazano na sliki 7.3. Označijo se lahko tudi



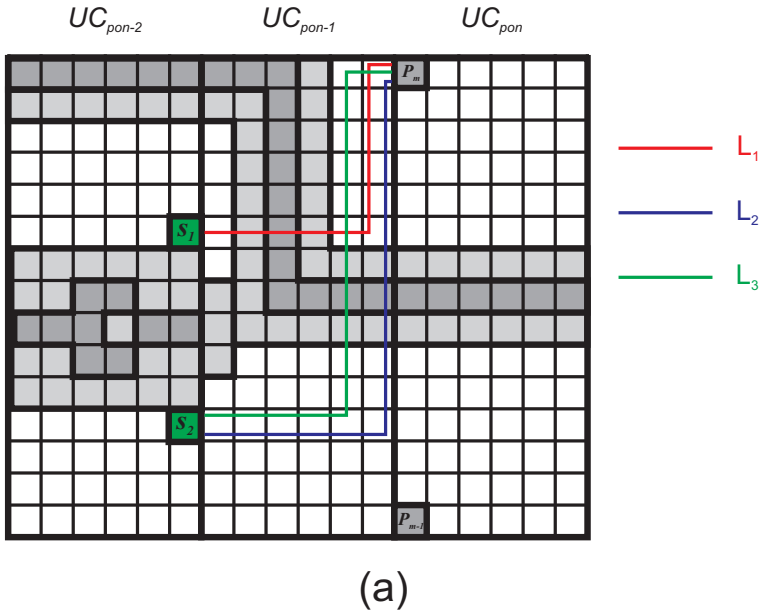
Slika 7.3 Označevanje področij z uporabo iskanja po labirintu v stopnji globalnega povezovanja. V vmesni coni, kot je UC_{izv+1}, se lahko označijo tudi zasedene celice, če pri tem lahko pride do križanja linij. Prosta področja na levem robu urine cone z logičnimi primitivi (UC_{izv+2}) so obarvana z rdečo barvo. Vsakemu prostemu področju na levem robu ustreza področje na desnem robu iste cone, tako da obe področji ležita na isti višini v mreži. Prosta področja na desnem robu cone so obarvana zeleno.

zasedene celice, kadar lahko pride do križanja. Ko označevanje doseže prosto področje p na levem robu urine cone z logičnimi primitivi, se p označi z ustrezno številko. Prosta področja na levem robu cone so na sliki 7.3 obarvana z rdečo. Označevanje v urini coni s primitivi ne poteka z algoritmom iskanja po labirintu, saj lahko skozi tako cono poteka le vodoravna linija. Zato se v trenutku, ko je označeno prosto področje p na levem robu cone UC_j, takoj označi področje p' na desnem robu UC_j, tako da p in p' ležita na isti višini v mreži. Razdalja med p in desnim robom UC_j je enaka $W-1$ področij, pri čemer je W širina cone. Če je $oznaka_p$ oznaka področja p, potem se področju p' pripiše oznaka $oznaka_{p'} = oznaka_p + W - 1$. Označevanje se nato nadaljuje od področja p' v UC_j proti coni UC_{pon}. Globalno povezovanje se zaključi, ko so označena vsa področja s ponornimi točkami.

Naslednja stopnja je podrobno povezovanje, pri katerem se določijo natančne lege linij, ki sestavljajo povezavo e_i. Podrobno povezovanje poteka v nasprotni smeri kot globalno, torej od cone UC_{pon} proti UC_{izv}. Najprej se poišče prostor za linijo v vmesni

coni UC_{pon-1}, ki povezuje ponor P_m na levem robu cone UC_{pon} in področje s, ki leži na desnem robu cone UC_{pon-2}. Za področje s je izbrano področje z najnižjo oznako med področji na desnem robu UC_{pon-2}. Obstaja lahko več področij z minimalno oznako, kot je prikazano na sliki 7.4(a), na kateri sta taki področji (s_1 in s_2) obarvani z zeleno barvo. Z algoritmom iskanja po liniji se poiščejo možni poteki linij med P_m in vsemi področji s_1, s_2, ... Tako določeni poteki linij imajo minimalno število kotov. Pogosto imajo tudi minimalno dolžino, še posebej na začetku povezovanja, ko še ni veliko področij zasedenih z že nameščenimi linijami. Na sliki 7.4(a) so prikazane tri linije in sicer rdeča L_1 med P_m in s_1, modra L_2 med P_m in s_2 ter zelena L_3 med P_m in s_2.

Vsaki liniji L_j se določi prioriteto $pr(L_j)$. Najnižjo prioriteto ima linija, pri kateri trojček *odmik*, d_1, d_2 ne zadošča načrtovalskim pravilom. To velja v primeru linije L_1. V primeru namestitve L_1 bi bilo potrebno vstaviti dodatno urino cono med UC_{pon-1} in UC_{pon} ter razdeliti L_1 med UC_{pon-1} in vstavljeno cono. To bi povečalo površino celotne fizične razmestitve strukture, vsoto dolžin vseh njenih linij in število urinih faz za izračun rezultata. V primeru namestitve L_2 ali L_3 ni potrebno vstaviti dodatne urine cone, zato velja $pr(L_1) < pr(L_2)$ in $pr(L_1) < pr(L_3)$. V naslednjem koraku se preštejejo vsa križanja linije L_j z ostalimi linijami. Linija z manjšim številom križanj ima višjo prioriteto kot linija z večjim številom križanj, če so pri obeh linijah upoštevana načrtovalska pravila. Vsaka od linij L_1, L_2 in L_3 na sliki 7.4(a) se enkrat križa z že nameščeno kotno linijo v UC_{pon-1}, zato ni mogoče določiti prioritete glede na število križanj, še vedno pa je $pr(L_1)$ nižja od $pr(L_2)$ in $pr(L_3)$. Nazadnje se določi prioriteta glede na bližino že nameščenih linij. Linija L_3 se nahaja bližje že nameščene linije kot L_2, zato velja $pr(L_2) < pr(L_3)$. Na ta način so linije nameščene bolj zgoščeno, kar pusti več prostora za namestitev povezav, ki sledijo e_i. Če ni prostega prostora za namestitev linije v coni UC_k, je potrebno vstaviti dodatno cono desno od UC_k. Tako se poveča prazen prostor, ki je na razpolago za namestitev linije. Ko so določene vse prioritete linij, se izbere linija z najvišjo prioriteto in se namesti v prosta področja. Najvišjo prioriteto na sliki 7.4(a) ima L_3, ki se namesti kot je prikazano na sliki 7.4(b). Če ima več kotnih linij v coni UC_k enake prioritete, se namesti kotna linija, katere navpični del je za dve področji odmaknjen od levega roba cone. Tako je med vsako QCA celico v navpičnem delu kotne linije in levim robom cone UC_k eno prosto področje. S tem je preprečena neželena interakcija med QCA celicami v navpičnem delu kotne linije v UC_k in izhodnimi QCA celicami v logičnih primitivih, ki ležijo na desnem robu cone UC_{k-1}. Navpični del kotne



Slika 7.4 Iskanje prostora za linijo med ponornimi točkami v UC_{pon} in področji na desnem robu UC_{pon-2}. Na sliki (a) sta dve ponorni točki P_m in P_{m-1} v UC_{pon}. Področji s_1 in s_2 na desnem robu UC_{pon-2} imata enako minimalno oznako, določeno v stopnji označevanja z algoritmom iskanja po labirintu. Z L_1, L_2 in L_3 so označeni možni poteki linij med P_m in s_1 oziroma s_2. Na sliki (b) je linija L_3 nameščena v zeleno obarvana področja. Namestitev L_3 določa področje s, ki ustreza področju s_2 na sliki (a). V modro obarvana področja se namesti linija, ki povezuje ponor P_{m-1} z linijo med P_m in s v zelenih področjih.

linije je lahko nameščen v področjih na desnem robu UC_k, vendar le takrat, ko se na njegovi desni strani v coni UC_{k+1} ne nahaja QCA celica s fiksno polarizacijo. Slednja vpliva na vse sosednje QCA celice. Za pravilno delovanje mora celica s fiksno polarizacijo vplivati le na sosednjo QCA celico v majoritetnih vratih, ne pa tudi na druge QCA celice. Zato mora biti med QCA celicami v navpičnem delu kotne linije v UC_k in celico s fiksno polarizacijo na levem robu UC_{k+1} vsaj eno prosto področje.

Z namestitvijo linije je določeno področje s, tako da linija povezuje P_m in s. V primeru na sliki 7.4(a) področju s ustreza s_2. V isti urini coni lahko obstaja več ponorov. Na sliki 7.4(b) se oba ponora P_{m-1} in P_m nahajata v coni UC_{pon}. Linija med P_m in s je že nameščena v zeleno obarvana področja. Potrebno je poiskati še linijo med P_{m-1} in s. Prostor za to linijo se določi na že opisan način, vendar pa sedaj obstaja še dodatni kriterij za prioriteto. Najprej se poiščejo možni poteki linij med P_{m-1} in s, nato se vsakemu določi prioriteta po opisanem postopku. Če ima na koncu postopka več potekov linij enako prioriteto, se izbere takšno linijo med P_{m-1} in s, ki se najhitreje priključi že nameščeni liniji med P_m in s. Tako se minimizira vsoto dolžin vseh linij v povezavi e_i in s tem tudi vsoto dolžin vseh linij v fizični razmestitvi strukture. Na sliki 7.4(b) so z modro barvo obarvana območja za linijo med P_{m-1} in že nameščeno linijo med P_m in s. Celotna linija v UC_{pon-1}, nameščena v zelenih in modrih področjih, ima vhod s in izhoda P_m in P_{m-1}, z njo pa se poleg prenosa signala izvede tudi njegova razvejitev.

Razvejitvena linija, ki povezuje izvorno točko z n_{ponor} ponornimi točkami, je sestavljena iz začetnega vodoravnega dela, navpičnega dela in n_{ponor} končnimi vodoravnimi deli, ki povezujejo navpični del s ponornimi točkami. Njena dolžina $d_{razvejitvena}$ je enaka vsoti

$$d_{razvejitvena} = d_1 + d_2 + n_{ponor} \cdot d_3, \tag{7.9}$$

kjer je d_1 dolžina začetnega vodoravnega dela, d_2 dolžina navpičnega dela in d_3 dolžina enega končnega vodoravnega dela. Pri tem je dolžina d_2 konstantna, za d_1 in d_3 pa velja $d_1 + d_3 + 1 = W(UC)$, kjer je $W(UC)$ širina urine cone. Čim manjša je dolžina d_3, tem manjša je skupna dolžina razvejitvene linije $d_{razvejitvena}$. Slednja je najmanjša pri vrednosti $d_3 = 0$, kar velja v primeru, ko se razvejitev signala izvede v področju na desnem robu cone. Zato se razvejitev signala poskuša realizirati čim bližje desnemu robu urine cone, pri čemer mora biti zadoščeno načrtovalskim pravilom. Če se v primeru na sliki 7.4(b) namesto linije L_3 namesti linija L_2, se razvejitev signala izvede v področju na desnem robu cone, načrtovalskim pravilom pa je še vedno zadoščeno. Zato se dejansko

namesti L_2.

Ko so nameščene vse linije v coni UC_{pon-1}, se povezovanje nadaljuje v UC_{pon-2}. Ker se v coni z logičnimi primitivi ne smejo nahajati kotne linije, se namesti vodoravno linijo med področjem s in področjem t na levem robu UC_{pon-2}. Takrat t postane nova ponorna točka, hkrati pa se lahko v UC_{pon-2} nahajajo še drugi ponori (P_{m-2}, P_{m-3}, ...). Po opisanem postopku se poiščejo linije v UC_{pon-3}, ki povezujejo ponore v UC_{pon-2} s področji na levem robu UC_{pon-4}. Algoritem se tako iterativno izvaja v conah UC_{pon}, UC_{pon-1}, ..., dokler ne doseže izvorne točke v UC_{izv}. Tedaj so nameščene vse linije, ki sestavljajo povezavo e_i. Celoten postopek nameščanja linij se nato izvede za povezave e_{i+1}, e_{i+2}, ..., e_{N_3}. Postopek povezovanja je povzet s psevdokodo v algoritmu 7.

Algoritem 7 Povezovanje logičnih primitivov v QCA.

```
procedure POVEZOVANJE
    for vsako povezavo e ∈ E do
     // določi področje z izvorom I in množico področij s ponori P = {P_1, ..., P_m}
        I = izvor(e); P = ponori(e)
        UC_izv = cona(I); UC_pon = cona(P_m);
     // globalno povezovanje z uporabo iskanja po labirintu
     // označuje področja, dokler niso označena vsa področja s ponori
        iskanje_po_labirintu()
     // podrobno povezovanje z uporabo iskanja po liniji
        for i = pon − 1 do i = izv + 1 do
           // določi področja za namestitev linije L_i v coni UC_i
           L_i = iskanje_po_liniji(UC_i)
        end for
    end for
// razdeljevanje kotnih linij, pri katerih parametri ne zadoščajo načrtovalskim pravilom
    for vsako kotno linijo kl do
        izračunaj parametre odmik, d_1, d_2
     // razdeljevanje linije med urine cone se izvede, če trojček odmik, d_1, d_2 ni veljaven
        if !veljavni_parametri(odmik, d_1, d_2) then
           // rekurzivno razdeljuj kl, dokler ni zadoščeno načrtovalskim pravilom
           razdeli(kl)
        end if
    end for
end procedure
```

8 Analiza rezultatov avtomatiziranega snovanja

8.1 Uvod

V pričujočem poglavju bomo predstavili fizične razmestitve struktur, izdelanih z uporabo metod za avtomatsko snovanje. Strukture bomo ovrednotili z definiranimi metrikami. Rezultate avtomatskega snovanja bomo primerjali z rezultati drugih avtorjev.

Zasnovane fizične razmestitve struktur temeljijo na lastnostih, opisanih v naslednjih alinejah:

- fizične razmestitve struktur so sestavljene iz polprevodnih QCA celic;
- širina in višina polprevodne QCA celice sta 18 nm;
- premer kvantne pike je 5 nm;
- razdalja med dvema kvantnima pikama v celici je 4 nm;
- razdalja med dvema QCA celicama je 2 nm [64, 5, 38, 6, 102, 39];
- dimenzije QCA celice določajo velikost posameznega področja, ki je enaka 20×20 nm;

- množico uporabljenih logičnih primitivov sestavljajo negatorji in nadgrajena majoritetna vrata;
- širina urine cone znaša šest področij;
- pravilno delovanje vseh zasnovanih struktur je verificirano z uporabo simulacijske metode koherenčnega vektorja v orodju QCADesigner.

8.2 Avtomatsko zasnovane strukture kvantnih celičnih avtomatov

Z uporabo predlaganih metod smo realizirali fizične razmestitve strukture enobitnega polnega seštevalnika, dvobitnega polnega seštevalnika in 4/1 multiplekserja. V nadaljevanju so predstavljene naštete strukture in opisan potek njihovega snovanja.

8.2.1 Struktura enobitnega polnega seštevalnika

Logično strukturo enobitnega polnega seštevalnika sestavljajo 3 vhodi, 2 izhoda, 3 majoritetna vrata in 2 negatorja. Logična shema enobitnega polnega seštevalnika je prikazana na sliki 8.1. Funkcijo enobitnega polnega seštevalnika $add(A, B, Cin) = (Sum, Cout)$ opisujeta izraza

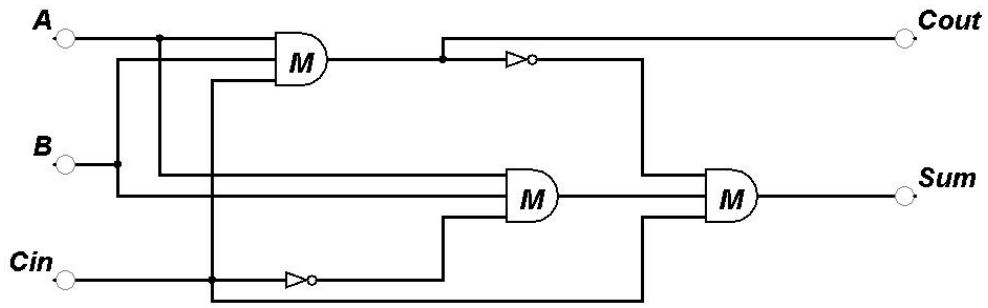
$$Sum = M(\overline{M(A, B, Cin)}, M(A, B, \overline{Cin}), Cin), \tag{8.1}$$

$$Cout = M(A, B, Cin), \tag{8.2}$$

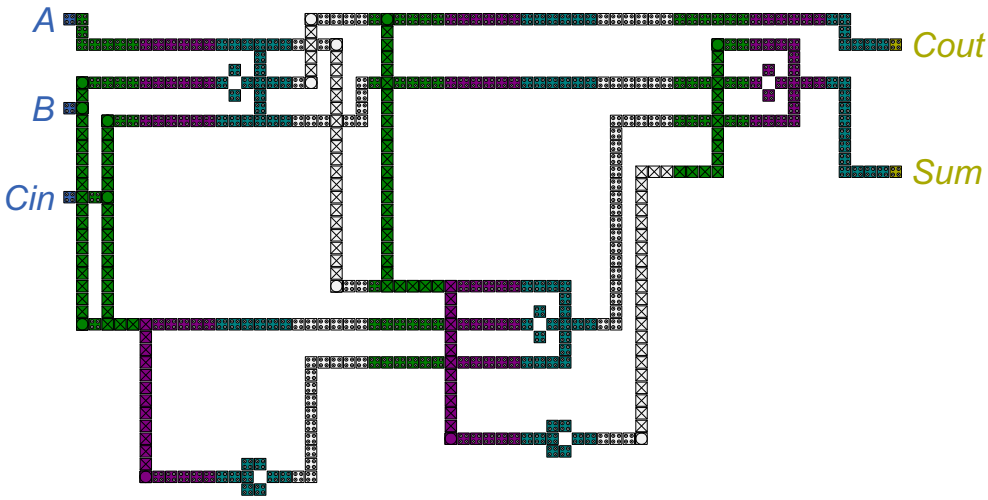
kjer M predstavlja majoritetno funkcijo.

V postopku iskanja fizične razmestitve strukture enobitnega polnega seštevalnika smo najprej določili začetno stanje razmestitve S_{01}, prikazano na sliki 8.2. Razmestitev je bila določena povsem naključno, tako da je z veliko verjetnostjo njena cena veliko višja od cene optimalne razmestitve. Za začetno razmestitev S_{01} je določena ocena vsote dolžin vseh linij $\sum_i N'_{dolzina_i} = 295$, ocena števila križanj linij $N'_{krizanje} = 8$ in ocena števila kotnih linij $N'_{kot} = 14$. Cena razmestitve se izračuna z uporabo enačbe (7.2) in je enaka $\text{cena}(S_{01}) = 295 + 5 \cdot 8 + 5 \cdot 14 = 405$.

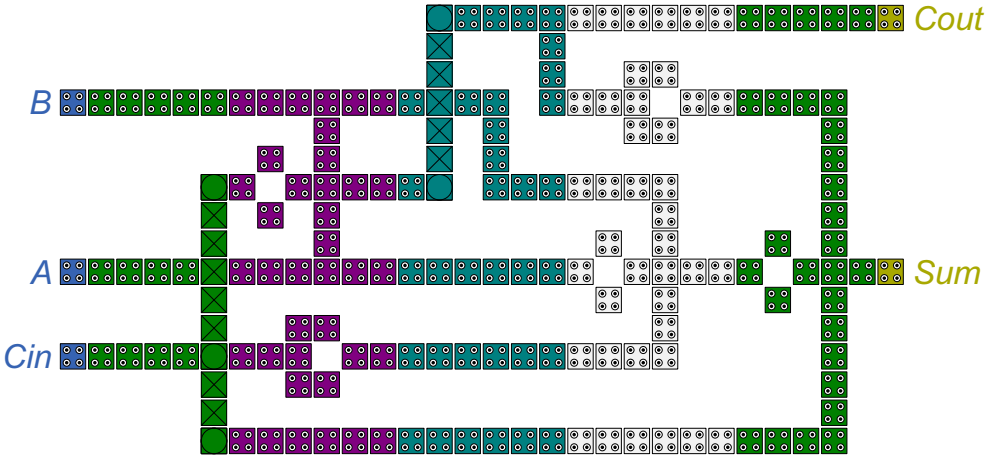
V nadaljevanju smo izvedli tri poskuse avtomatskega iskanja. V vsakem poskusu je bilo izvedenih 1 000 000 iteracij. Višina posamezne urine cone je znašala 50 področij. Začetno stanje je vsakokrat določala razmestitev S_{01}. Rezultati poskusov so fizične razmestitve S_{11}, S_{21} in S_{31}, prikazane na slikah 8.3, 8.4 in 8.5.



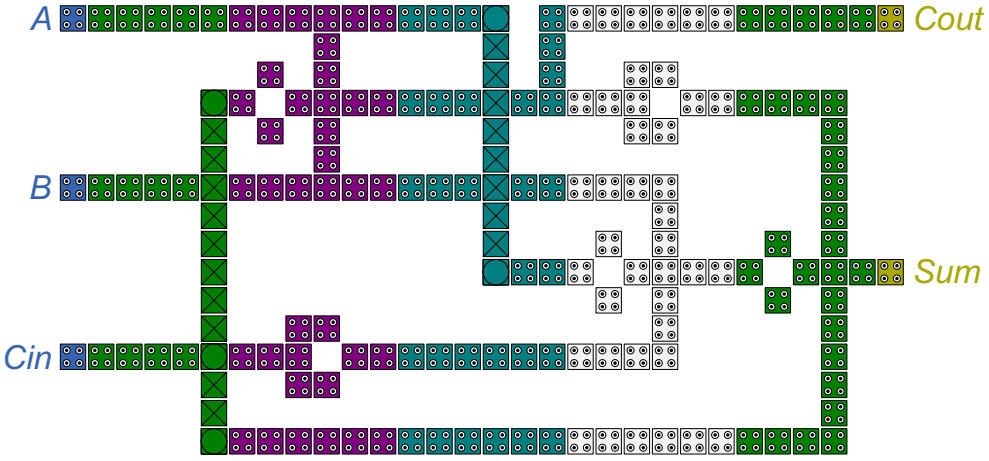
Slika 8.1 Logična shema enobitnega polnega seštevalnika.



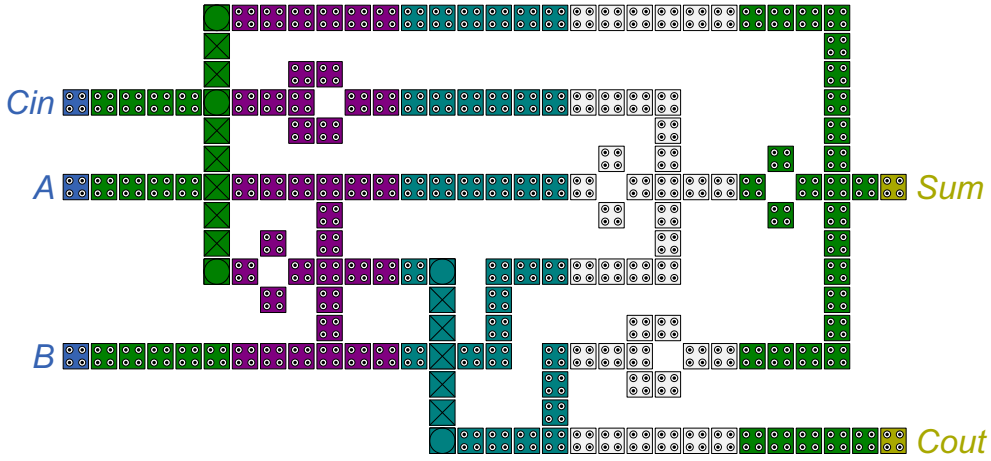
Slika 8.2 Začetna razmestitev S_{01} pri iskanju fizične razmestitve strukture enobitnega polnega seštevalnika.



Slika 8.3 Rezultat prvega poskusa iskanja fizične razmestitve strukture enobitnega polnega seštevalnika pri začetni razmestitvi S_{01} je razmestitev S_{11}.



Slika 8.4 Rezultat drugega poskusa iskanja fizične razmestitve strukture enobitnega polnega seštevalnika pri začetni razmestitvi S_{01} je razmestitev S_{21}.



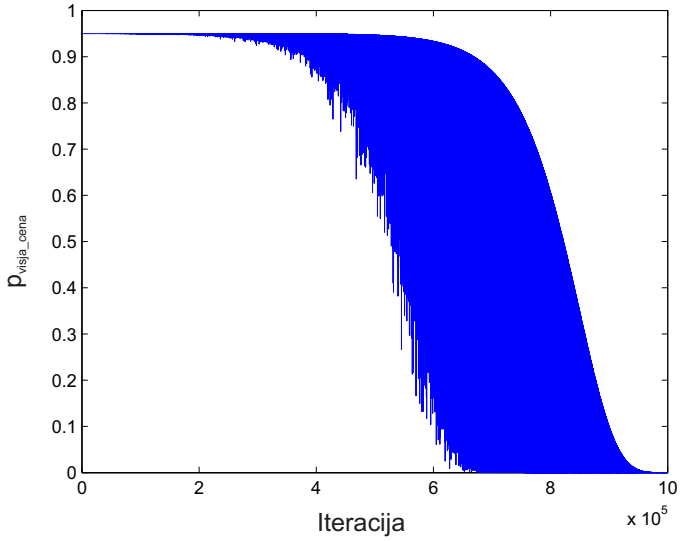
Slika 8.5 Rezultat tretjega poskusa iskanja fizične razmestitve strukture enobitnega polnega seštevalnika pri začetni razmestitvi S_{01} je razmestitev S_{31}.

Za posamezno razmestitev veljajo ocene:

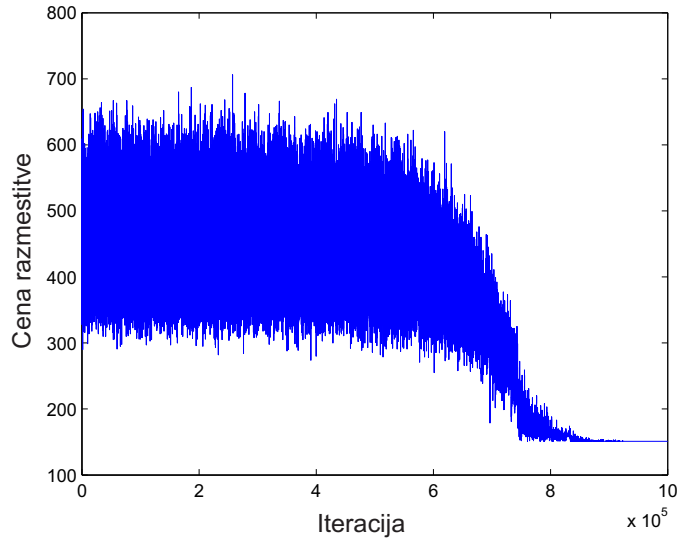
- razmestitev S_{11}: $\sum_i N'_{dolzina_i} = 111$, $N'_{krizanje} = 3$, $N'_{kot} = 5$, cena$(S_{11}) = 151$, algoritem je našel razmestitev v iteraciji z indeksom 760 699;
- razmestitev S_{21}: $\sum_i N'_{dolzina_i} = 114$, $N'_{krizanje} = 4$, $N'_{kot} = 4$, cena$(S_{21}) = 154$, algoritem je našel razmestitev v iteraciji z indeksom 990 124;
- razmestitev S_{31}: $\sum_i N'_{dolzina_i} = 111$, $N'_{krizanje} = 3$, $N'_{kot} = 5$, cena$(S_{31}) = 151$, algoritem je našel razmestitev v iteraciji z indeksom 756 293.

Na sliki 8.6(a) je prikazan graf verjetnosti prehoda v stanje z višjo ceno v odvisnosti od indeksa iteracije. Verjetnost prehoda v stanje z višjo ceno p_{visja_cena} se izračuna z enačbo (7.5), pri čemer ima novo stanje višjo ceno od sedanjega stanja, torej velja cena$(s_i) >$ cena(s_{i-1}). Verjetnost prehoda v stanje z nižjo ceno je po enačbi (7.5) vedno enaka 1. Ko je vrednost p_{visja_cena} blizu 1, je iskanje zelo naključno, saj se poleg gotovih prehodov v stanja z nižjo ceno pogosto izvede tudi prehod v stanje z višjo ceno. Ko pa se vrednost p_{visja_cena} bliža 0, je iskanje vse bolj požrešno, ker se prehodi v stanja z višjo ceno skoraj ne dogodijo.

Slika 8.6(b) prikazuje graf odvisnosti cene razmestitve glede na indeks iteracije. Grafa na sliki 8.6 prikazujeta podatke, zbrane v prvem poskusu pri iskanju razmestitve S_{11}. Podatke iz drugega poskusa prikazujeta grafa na sliki 8.7, podatke iz tretjega poskusa

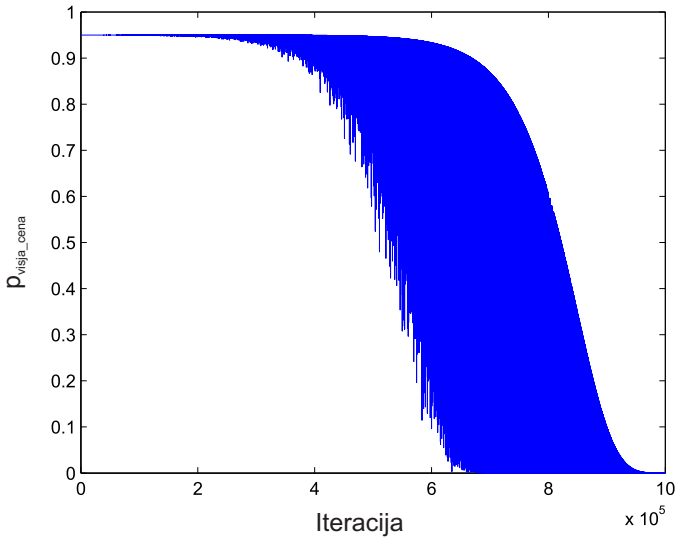


(a)

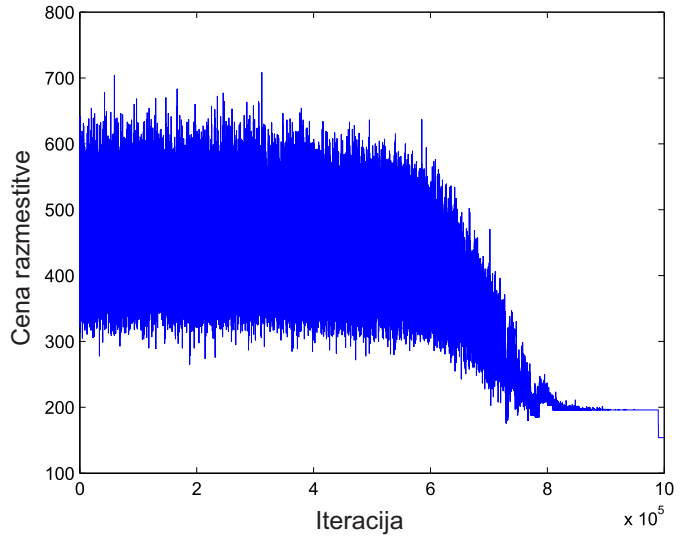


(b)

Slika 8.6 Graf odvisnosti verjetnosti prehoda v stanje z višjo ceno glede na indeks iteracije (a). Graf odvisnosti cene razmestitve glede na indeks iteracije (b). Podatki so bili zbrani v prvem poskusu iskanja fizične razmestitve strukture enobitnega polnega seštevalnika, pri čemer je bila rezultat razmestitev S_{11}. Najnižja cena razmestitve je bila določena v iteraciji z indeksom 760 699.

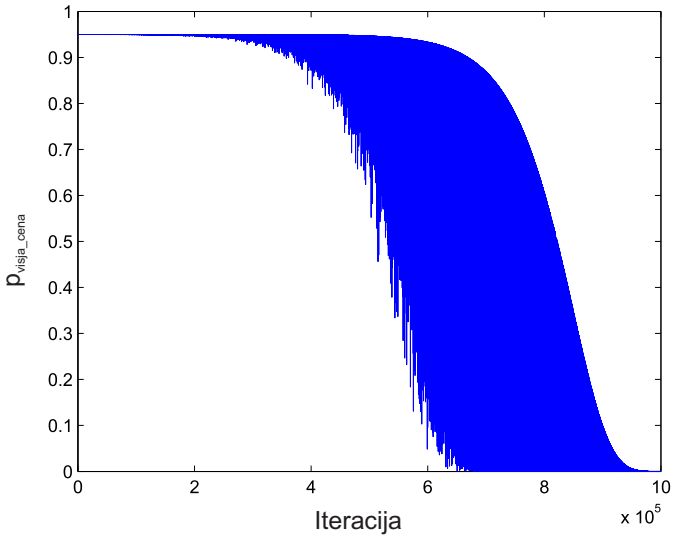


(a)

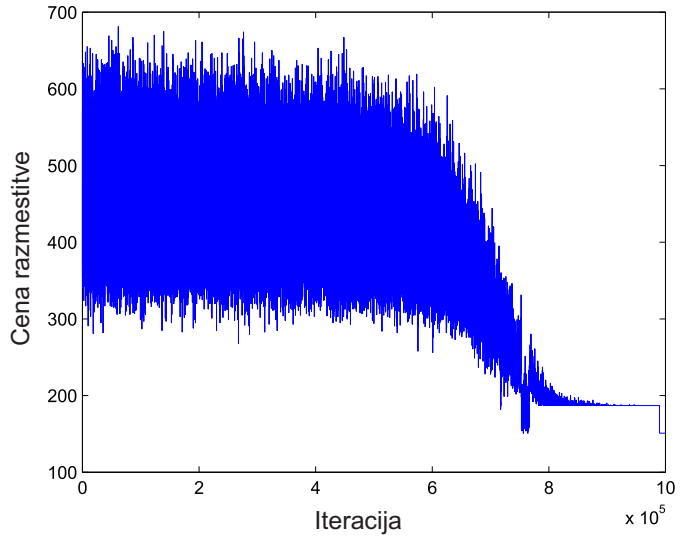


(b)

Slika 8.7 Graf odvisnosti verjetnosti prehoda v stanje z višjo ceno glede na indeks iteracije (a). Graf odvisnosti cene razmestitve glede na indeks iteracije (b). Podatki so bili zbrani v drugem poskusu iskanja fizične razmestitve strukture enobitnega polnega seštevalnika, pri čemer je bila rezultat razmestitev S_{21}. Najnižja cena razmestitve je bila določena v iteraciji z indeksom 990 124, torej šele v zadnjem delu izvajanja algoritma pri lokalni optimizaciji s požrešnim iskanjem.



(a)



(b)

Slika 8.8 Graf odvisnosti verjetnosti prehoda v stanje z višjo ceno glede na indeks iteracije (a). Graf odvisnosti cene razmestitve glede na indeks iteracije (b). Podatki so bili zbrani v tretjem poskusu iskanja fizične razmestitve strukture enobitnega polnega seštevalnika, pri čemer je bila rezultat razmestitev S_{31}. Najnižja cena razmestitve je bila določena v iteraciji z indeksom 756 293.

pa grafa na sliki 8.8.

V prvem poskusu je algoritem našel razmestitev S_{11} z najnižjo ceno v iteraciji z indeksom 760 699. V nadaljnjem iskanju se cena razmestitve ni znatno spreminjala. Kot je razvidno iz grafa 8.6(b), se je cena ustalila v minimumu v bližini iteracije z indeksom 900 000. Do iteracije z indeksom 990 000 se cena ni spreminjala. V zadnjih 10 000 iteracijah je algoritem požrešno optimiziral razmestitev S_{11}, vendar se njena cena do zaključka iskanja ni spremenila. V drugem poskusu je algoritem našel razmestitev S'_{21} pred iteracijo z indeksom 800 000. Od iteracije z indeksom 990 000 je algoritem požrešno optimiziral do tedaj najcenejšo razmestitev S'_{21} in v njeni bližini našel lokalni minimum S_{21} v iteraciji z indeksom 990 124. Tako je končna rešitev postala razmestitev S_{21} z najnižjo ceno. V tretjem poskusu je algoritem našel razmestitev S_{31} v iteraciji z indeksom 756 293. Kasneje se je izvedel prehod v stanje S'_{31} z višjo ceno od S_{31}. Stanje S'_{31} je ležalo v nekem lokalnem minimumu, iz katerega iskanje ni ušlo zaradi majhne verjetnosti prehoda v stanje z višjo ceno v zadnjem delu izvajanja algoritma. Vendar pa je algoritem v iteraciji z indeksom 990 000 izbral najcenejšo do tedaj najdeno razmestitev S_{31} in jo poskušal požrešno optimizirati v zadnjih 10 000 iteracijah. Njena cena se ni spremenila in končna rešitev je postala razmestitev S_{31}.

Že v primeru enostavnih struktur je prostor možnih stanj izredno velik. V usmerjenem grafu logične sheme s slike 8.1 vsi logični primitivi ležijo na kritičnih poteh. Vsaka kritična pot vsebuje 3 logične primitive, zato ima fizična razmestitev strukture enobitnega polnega seštevalnika 3 urine cone z logičnimi primitivi UC_{L1}, UC_{L2} in UC_{L3}. Cona UC_{L1} vsebuje majoritetne vrata in negator, tudi cona UC_{L2} vsebuje majoritetne vrata in negator, UC_{L3} pa vsebuje ena majoritetna vrata. Prva urina cona UC_1 vsebuje 3 vhodne priključke, zadnja cona UC_n pa vsebuje 2 izhodna priključka. Širina vsake cone je enaka 6 področij, višina pa naj bo enaka 20 področij. Vhodni priključki so lahko nameščeni le na levem robu UC_1, izhodni priključki pa le na desnem robu UC_n. Vsi logični primitivi imajo enako širino kot urina cona. Zato so x koordinate vhodno/izhodnih priključkov in logičnih primitivov fiksne, spreminja pa se lahko njihova y koordinata znotraj cone. Prvi vhodni priključek A se lahko namesti na 20 različnih področij. Naslednji vhodni priključek B ne more zasedati področja, na katerem je nameščen A. Prav tako mora biti od A oddaljen vsaj za dve področji, zato ne more zasedati področja nad in pod A. Tako za namestitev priključka B preostane 20-3=17 področij. Za priključek Cin preostane še 17-3=14 področij. Če se na primer priključek A nahaja povsem na spodnjem robu cone,

je možnosti za namestitev ostalih priključkov še več, saj se med zasedena področja ne prišteva področje, ki se nahaja pod A, ker leži izven urine cone. Tako je spodnja meja za število razmestitev priključkov v UC_1 enaka $R_{UC_1} = 20 \cdot 17 \cdot 14 = 4760$. Največja višina nadgrajenih majoritetnih vrat (slika 4.4) znaša 13 področij. Zato se lahko majoritetna vrata v UC_{L1} namestijo na 20-13+1=8 načinov. Ker morata biti področji pod in nad majoritetnimi vrati prosti, za namestitev negatorja z višino 3 področij preostanejo še 8-2-3+1=4 mesta. Navpična dela kotnih linij v nadgrajenih majoritetnih vratih sta lahko podaljšana za 0, 1, 2 ali 3 celice, kar določa 4 različne možne višine nadgrajenih majoritetnih vrat. Tako je spodnja meja za razmestitev logičnih primitivov v UC_{L1} enaka $R_{UC_{L1}} = 8 \cdot 4 \cdot 4 = 128$. Enako velja za cono UC_{L2}, torej $R_{UC_{L2}} = 8 \cdot 4 \cdot 4 = 128$. V UC_{L3} se lahko nadgrajena majoritetna vrata namestijo na najmanj $R_{UC_{L3}} = 8 \cdot 4 = 32$ načinov. Izhodna priključka v UC_n lahko tvorita vsaj $R_{UC_1} = 20 \cdot 17 = 340$ različnih razmestitev. Število vseh možnih razmestitev vhodno/izhodnih priključkov in logičnih primitivov strukture enobitnega polnega seštevalnika v conah z višino 20 področij torej presega $R_{UC_1} \cdot R_{UC_{L1}} \cdot R_{UC_{L2}} \cdot R_{UC_{L3}} \cdot R_{UC_n} > 848 \cdot 10^9$, zato bi bilo izčrpno preiskovanje vseh stanj časovno zelo potratno.

Na začetku algoritem simuliranega ohlajanja, opisan v razdelku 7.3, naključno preiskuje celoten prostor stanj. Verjetnost prehoda v stanje z višjo ceno p_{visja_cena} je precej velika, tako da se iskanje ne ujame takoj v lokalnem minimumu. Ker je veliko prehodov v stanja z višjo ceno, na začetku delovanja algoritma cena razmestitve zelo niha. Po polovici izvedenih iteracij začne p_{visja_cena} hitro padati. Slabša stanja imajo manjšo možnost izbire, zato cena razmestitve prične padati. Optimizacija postaja vse bolj lokalna, tako da je razlika med cenama dveh zaporednih stanj vse manjša. V zadnjem delu izvajanja se p_{visja_cena} približa 0. To pomeni, da se izvaja požrešno iskanje, pri katerem je prehod možen le v stanja z nižjo ceno. Algoritem poskuša lokalno optimizirati najcenejše stanje, ki je bilo določeno v eni izmed predhodnih iteracij. To stanje se približuje nekemu lokalnemu minimumu, ki pa ima precej nižjo ceno kot lokalni minimumi, ki jih je algoritem obiskal na začetku izvajanja.

Ker ima lahko več različnih rešitev enako ceno, lahko algoritem najde različne rešitve pri isti začetni razmestitvi. Vsaka rešitev predstavlja približek lokalnega minimuma s precej nižjo ceno, kot jo ima neoptimizirano začetno stanje. Ker algoritem temelji na naključnih potezah za spremembo stanja, lahko pri isti začetni razmestitvi najde različne rešitve. Če bi želeli z gotovostjo najti globalni minimum, bi moral algoritem pregledati

vsa možna stanja, za kar bi bilo potrebno izjemno veliko število iteracij.

Naključno smo določili tudi začetno razmestitev S_{02}, ki se razlikuje od razmestitve S_{01}, njena cena pa je še vedno veliko višja od cen optimiziranih rešitev. Razmestitev S_{02} je prikazana na sliki 8.9. Za začetno razmestitev S_{02} je določena ocena vsote dolžin vseh linij $\sum_i N'_{dolzina_i} = 410$, ocena števila križanj linij $N'_{krizanje} = 13$ in ocena števila kotnih linij $N'_{kot} = 15$. Cena razmestitve je enaka $\text{cena}(S_{02}) = 410 + 5 \cdot 13 + 5 \cdot 15 = 550$.

Nato smo ponovno izvedli tri poskuse avtomatskega iskanja. V vsakem poskusu je bilo izvedenih 1 000 000 iteracij. Višina posamezne urine cone je znašala 50 področij. Začetno stanje je vsakokrat določala razmestitev S_{02}. Rezultati poskusov so fizične razmestitve S_{12}, S_{22} in S_{32}, prikazane na slikah 8.10, 8.11 in 8.12.

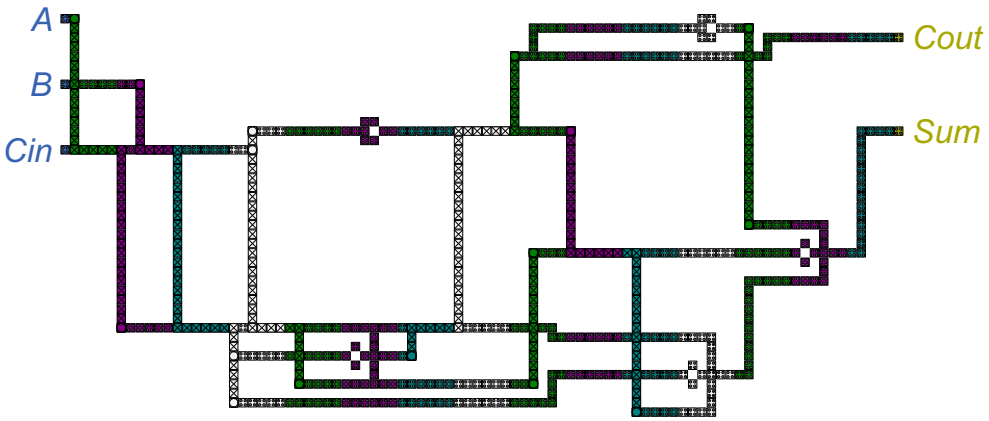
Za posamezno razmestitev veljajo ocene:

- razmestitev S_{12}: $\sum_i N'_{dolzina_i} = 114$, $N'_{krizanje} = 3$, $N'_{kot} = 4$, $\text{cena}(S_{12}) = 149$, algoritem je našel razmestitev v iteraciji z indeksom 754 801;
- razmestitev S_{22}: $\sum_i N'_{dolzina_i} = 111$, $N'_{krizanje} = 3$, $N'_{kot} = 5$, $\text{cena}(S_{22}) = 151$, algoritem je našel razmestitev v iteraciji z indeksom 760 487;
- razmestitev S_{32}: $\sum_i N'_{dolzina_i} = 114$, $N'_{krizanje} = 3$, $N'_{kot} = 5$, $\text{cena}(S_{32}) = 154$, algoritem je našel razmestitev v iteraciji z indeksom 990 217.

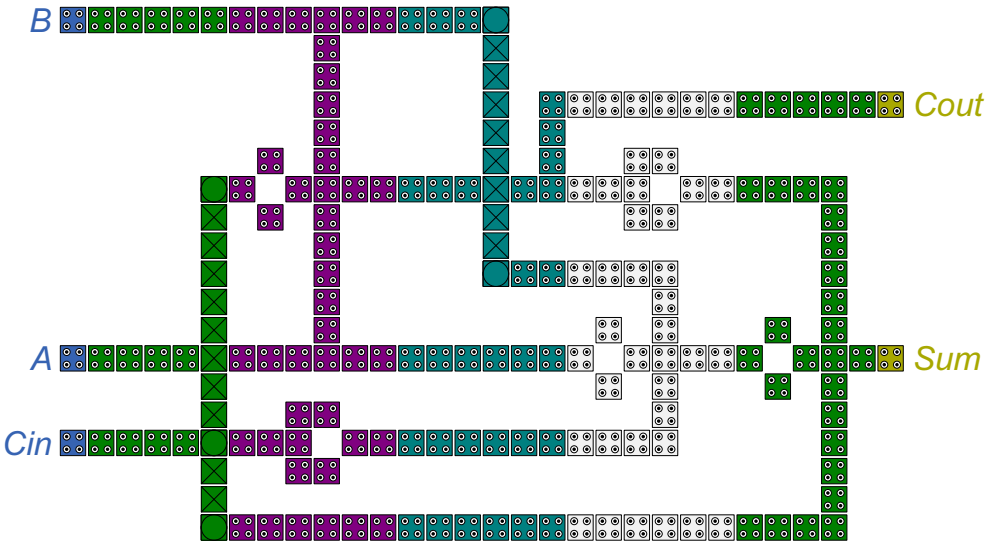
Rešitve S_{11}, S_{21} in S_{31} izhajajo iz začetne razmestitve S_{01}, rešitve S_{12}, S_{22} in S_{32} pa izhajajo iz začetne razmestitve S_{02}. Čeprav se začetni razmestitvi S_{01} in S_{02} med seboj zelo razlikujeta, imajo vse rešitve podobno ceno. To omogoča algoritem simuliranega ohlajanja, ki lahko pri iskanju uide iz lokalnega minimuma. Če bi se iskanje ustavilo v prvem lokalnem minimumu, bi se dve rešitvi, ki izhajata iz dveh različnih začetnih razmestitev, med seboj lahko zelo razlikovali.

Vrednosti optimizacijskih kriterijev naključno določenih začetnih razmestitev S_{01} in S_{02} so navedene v tabeli 8.1. Vrednosti optimiziranih kriterijev fizičnih razmestitev struktur S_{11}, S_{21}, S_{31}, S_{12}, S_{22} in S_{32} so navedene v tabeli 8.2.

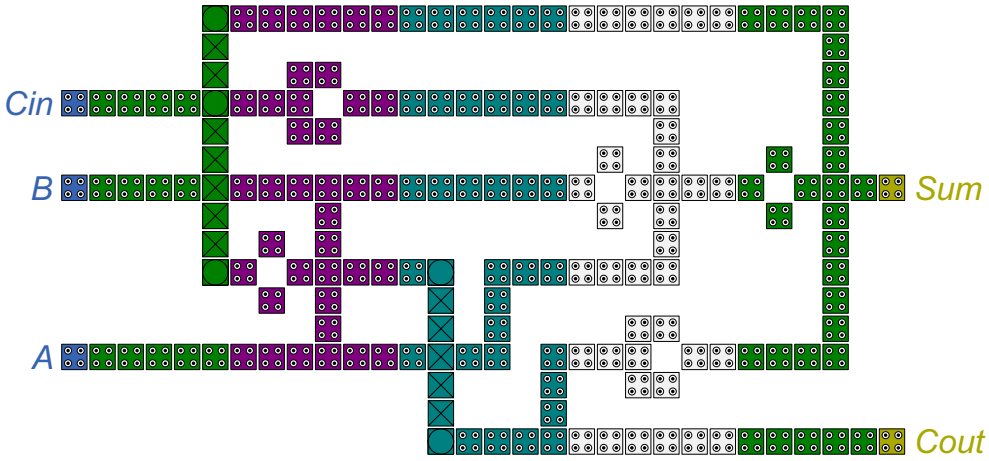
Pravilno delovanje avtomatsko zasnovanih struktur potrjuje grafični prikaz rezultatov simulacije, pridobljenih z uporabo verifikacijskega orodja QCADesigner, na sliki 8.13. Rezultati simulacij so enaki za vse fizične razmestitve struktur S_{11}, S_{21}, S_{31}, S_{12}, S_{22} in S_{32}.



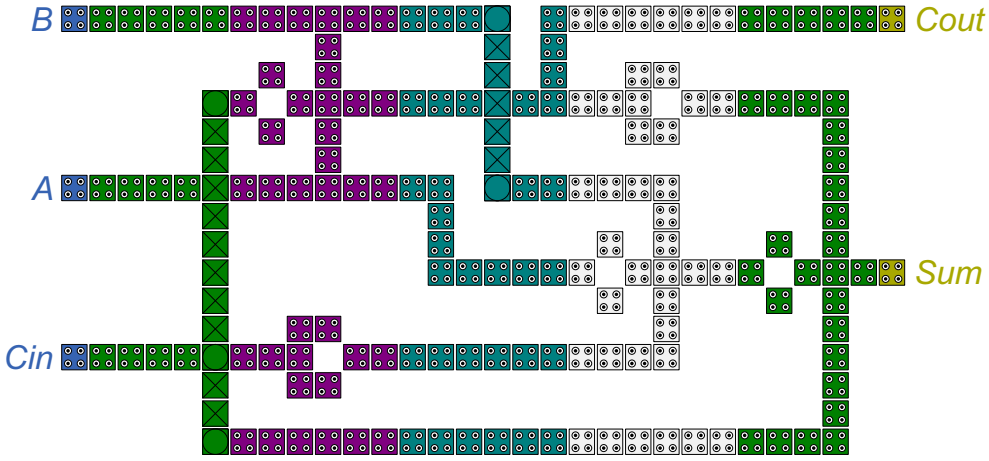
Slika 8.9 Začetna razmestitev S_{02} pri iskanju fizične razmestitve strukture enobitnega polnega seštevalnika.



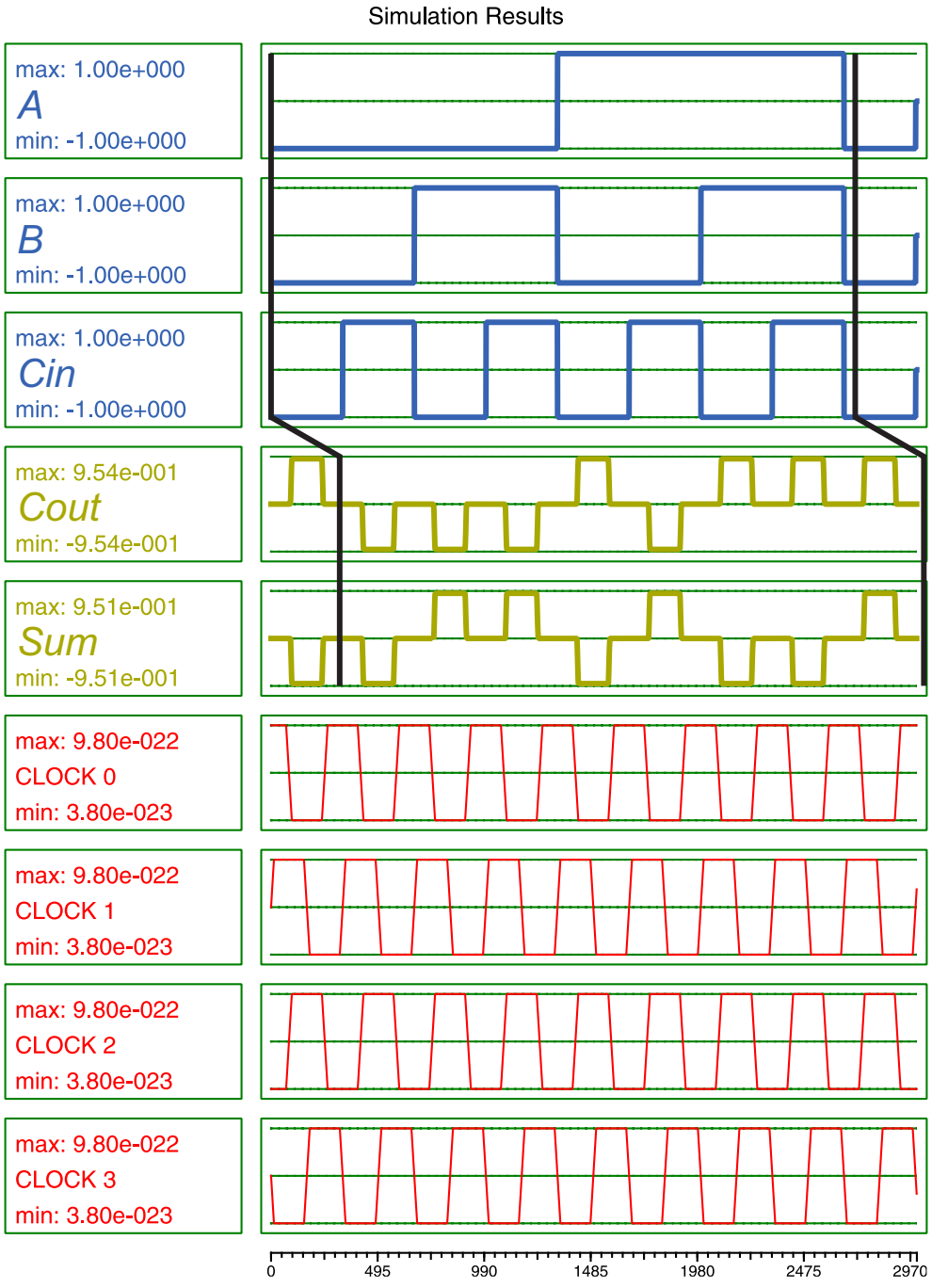
Slika 8.10 Rezultat prvega poskusa iskanja fizične razmestitve strukture enobitnega polnega seštevalnika pri začetni razmestitvi S_{02} je razmestitev S_{12}.



Slika 8.11 Rezultat drugega poskusa iskanja fizične razmestitve strukture enobitnega polnega seštevalnika pri začetni razmestitvi S_{02} je razmestitev S_{22}.



Slika 8.12 Rezultat tretjega poskusa iskanja fizične razmestitve strukture enobitnega polnega seštevalnika pri začetni razmestitvi S_{02} je razmestitev S_{32}.



Slika 8.13 Grafični prikaz rezultatov simulacije delovanja avtomatsko zasnovane fizične razmestitve strukture enobitnega polnega seštevalnika. Zakasnitev signala v strukturi znaša 5 urinih faz oziroma 1,25 urinega cikla.

kriterij	S_{01}	S_{02}
število urinih faz	11	15
vsota dolžin vseh linij	449	601
število kotnih linij	28	38
porabljena površina	66×38	90×43
število križanj linij	10	13

Tabela 8.1 Vrednosti optimizacijskih kriterijev naključno določenih začetnih razmestitev S_{01} in S_{02}.

kriterij	S_{11}	S_{21}	S_{31}	S_{12}	S_{22}	S_{32}
število urinih faz	5	5	5	5	5	5
vsota dolžin vseh linij	104	107	104	107	104	107
število kotnih linij	7	5	7	5	7	7
porabljena površina	30×16	30×16	30×16	30×19	30×16	30×16
število križanj linij	2	3	2	2	2	2

Tabela 8.2 Vrednosti optimiziranih kriterijev avtomatsko zasnovanih fizičnih razmestitev struktur S_{11}, S_{21}, S_{31}, S_{12}, S_{22} in S_{32}.

Za primerjavo med avtomatsko zasnovanimi in že obstoječimi strukturami enobitnega polnega seštevalnika so v tabeli 8.3 navedene vrednosti kriterijev za nekatere že zasnovane strukture drugih avtorjev.

kriterij	[27]	[5]	[89]
število urinih faz	3	5	16
vsota dolžin vseh linij	162	120	268
število kotnih linij	6	5	13
porabljena površina	17×25	20×22	56×63
število križanj linij	9	3	0

Tabela 8.3 Vrednosti optimizacijskih kriterijev fizičnih razmestitev struktur enobitnega polnega seštevalnika, ki so delo drugih avtorjev.

Slednje so opisane v naslednjih alinejah:

- Struktura v delu [27]: Razdeljena je na tri pravokotne urine cone z enakimi višinami, medtem ko se njihove širine med seboj razlikujejo. Križanje linij je implementirano

na istem nivoju z uporabo navadnih in rotiranih QCA celic.

- Struktura v delu [5]: Urine cone nimajo pravilnih oblik. Nekatere cone vsebujejo samo eno QCA celico.
- Struktura v delu [89]: Enobitni seštevalnik temelji na urinih conah kvadratne oblike s stranico velikosti 7 področij. Avtorji so križanja linij odpravili s podvojitvijo vseh treh vhodnih priključkov.

8.2.2 Struktura dvobitnega polnega seštevalnika

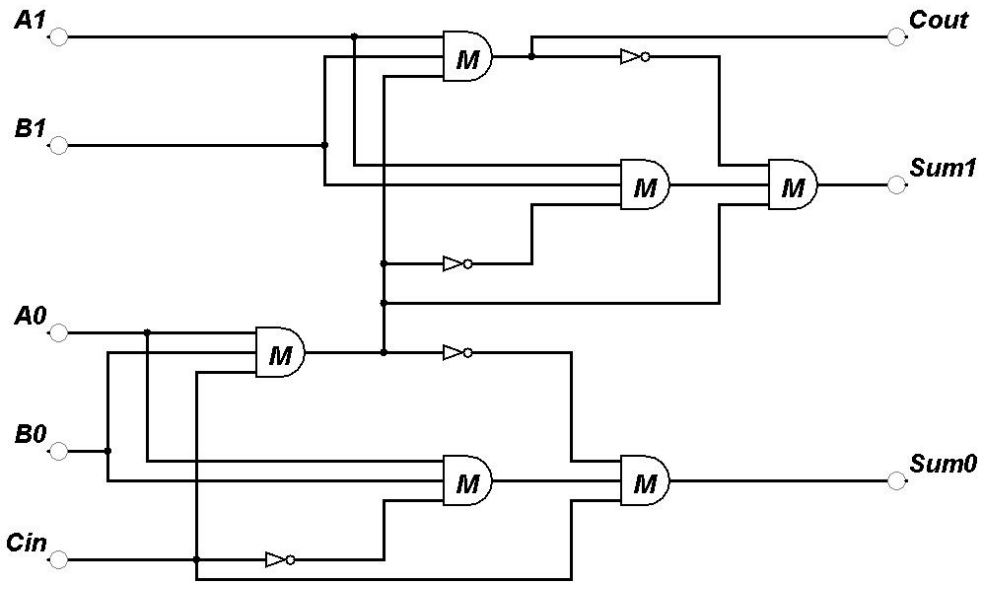
Struktura dvobitnega polnega seštevalnika omogoča seštevanje dveh dvobitnih števil A in B ter prenosa Cin. Rezultat sta dvobitna vsota in prenos $Cout$. Spremenljivke so povezane z enačbami

$$A = 2 \times A1 + A0, \tag{8.3}$$

$$B = 2 \times B1 + B0, \tag{8.4}$$

$$A + B + Cin = 4 \times Cout + 2 \times Sum1 + Sum0. \tag{8.5}$$

Logična shema dvobitnega polnega seštevalnika je prikazana na sliki 8.14.



Slika 8.14 Logična shema dvobitnega polnega seštevalnika.

Pri iskanju fizične razmestitve strukture dvobitnega polnega seštevalnika smo najprej določili začetno stanje razmestitve T_{01}, prikazano na sliki 8.15. Razmestitev je bila določena naključno, tako da je njena cena zelo verjetno veliko višja od cene optimalne razmestitve. Za začetno razmestitev T_{01} je določena ocena vsote dolžin vseh linij $\sum_i N'_{dolzina_i} = 1015$, ocena števila križanj linij $N'_{krizanje} = 49$ in ocena števila kotnih linij $N'_{kot} = 28$. Cena razmestitve se izračuna z uporabo enačbe (7.2) in je enaka cena(T_{01}) = 1400.

Nato smo izvedli tri poskuse avtomatskega iskanja, tako da je bila začetna razmestitev vedno T_{01}. V vsakem poskusu je bilo izvedenih 1 000 000 iteracij. Višina posamezne urine cone je znašala 100 področij. Rezultati poskusov so fizične razmestitve T_{11}, T_{21} in T_{31}, prikazane na slikah 8.16, 8.17 in 8.18.

Za posamezno razmestitev veljajo ocene:

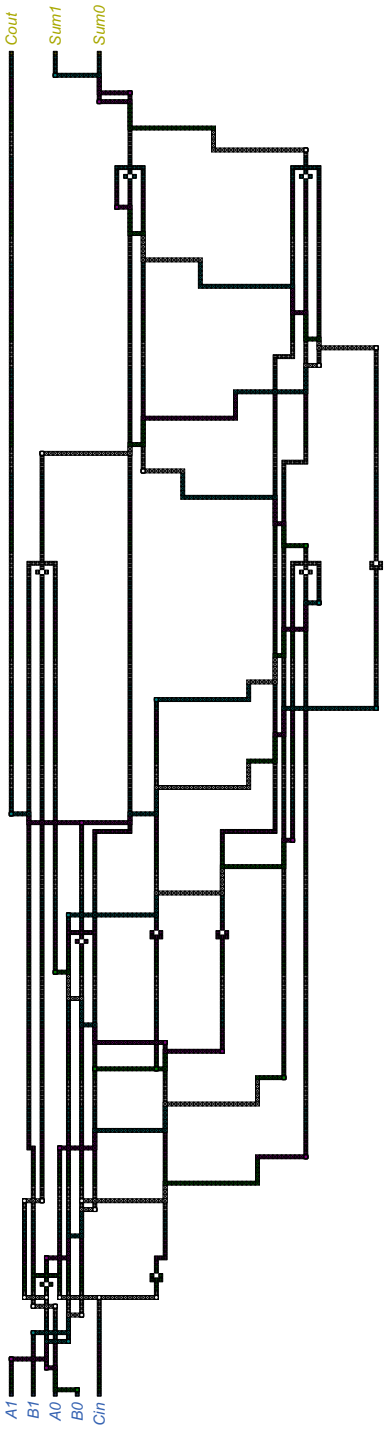
- razmestitev T_{11}: $\sum_i N'_{dolzina_i} = 258$, $N'_{krizanje} = 7$, $N'_{kot} = 13$, cena(T_{11}) = 358, algoritem je našel razmestitev v iteraciji z indeksom 797 341;
- razmestitev T_{21}: $\sum_i N'_{dolzina_i} = 271$, $N'_{krizanje} = 9$, $N'_{kot} = 11$, cena(T_{21}) = 371, algoritem je našel razmestitev v iteraciji z indeksom 750 070;
- razmestitev T_{31}: $\sum_i N'_{dolzina_i} = 280$, $N'_{krizanje} = 8$, $N'_{kot} = 10$, cena(T_{31}) = 370, algoritem je našel razmestitev v iteraciji z indeksom 811 808.

V drugem delu postopka iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika smo določili novo začetno razmestitev T_{02}. Slednja se razlikuje od razmestitve T_{01}, obe pa sta bili določeni naključno. Razmestitev T_{02} je prikazana na sliki 8.19. Za začetno razmestitev T_{02} je določena ocena vsote dolžin vseh linij $\sum_i N'_{dolzina_i} = 828$, ocena števila križanj linij $N'_{krizanje} = 48$ in ocena števila kotnih linij $N'_{kot} = 29$. Cena razmestitve je enaka cena(T_{02}) = 1213.

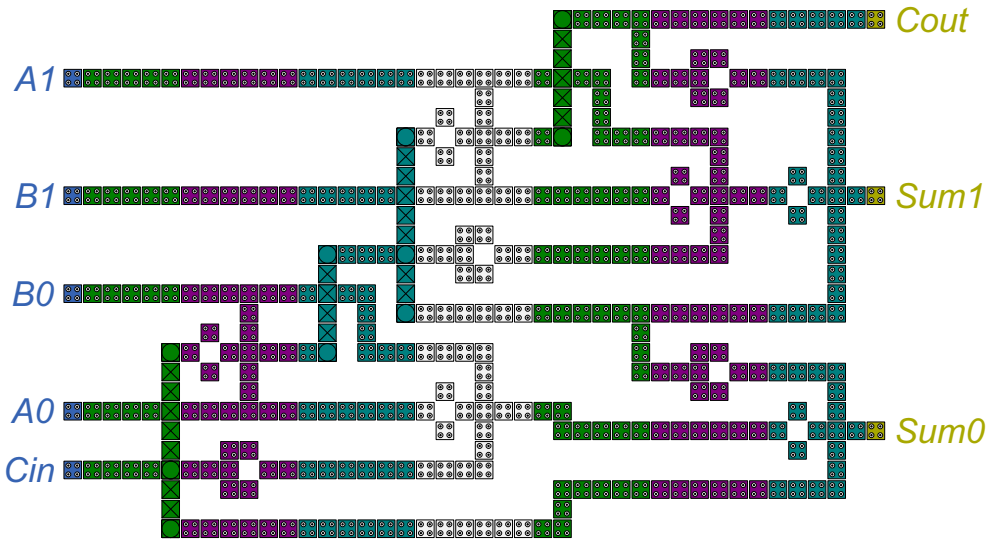
Nato smo ponovno izvedli tri poskuse avtomatskega iskanja. V vsakem poskusu je bilo izvedenih 1 000 000 iteracij. Višina posamezne urine cone je znašala 100 področij. Začetno stanje je vsakokrat določala razmestitev T_{02}. Rezultati poskusov so fizične razmestitve T_{12}, T_{22} in T_{32}, prikazane na slikah 8.20, 8.21 in 8.22.

Za posamezno razmestitev veljajo ocene:

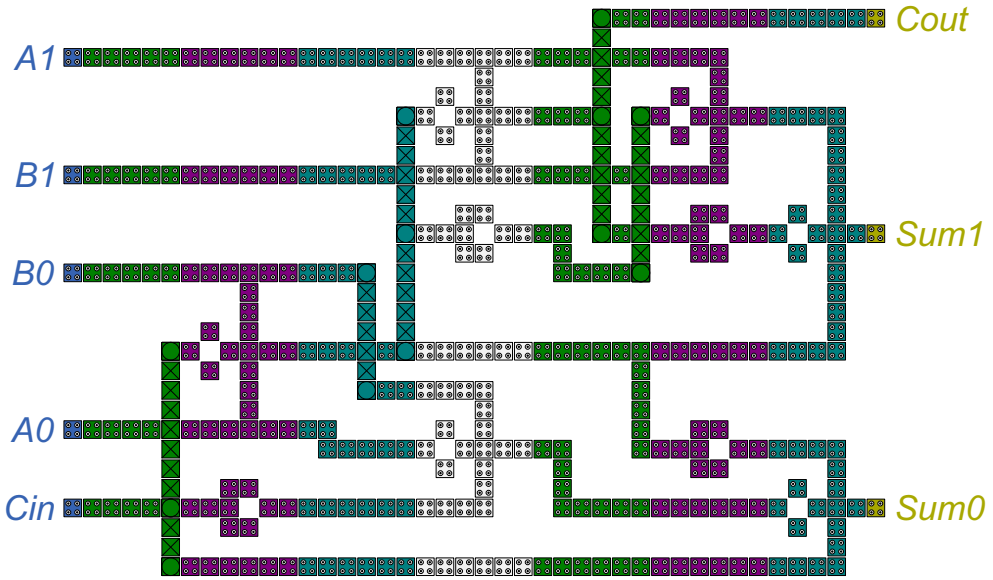
- razmestitev T_{12}: $\sum_i N'_{dolzina_i} = 260$, $N'_{krizanje} = 7$, $N'_{kot} = 12$, cena(T_{12}) = 355, algoritem je našel razmestitev v iteraciji z indeksom 802 858;



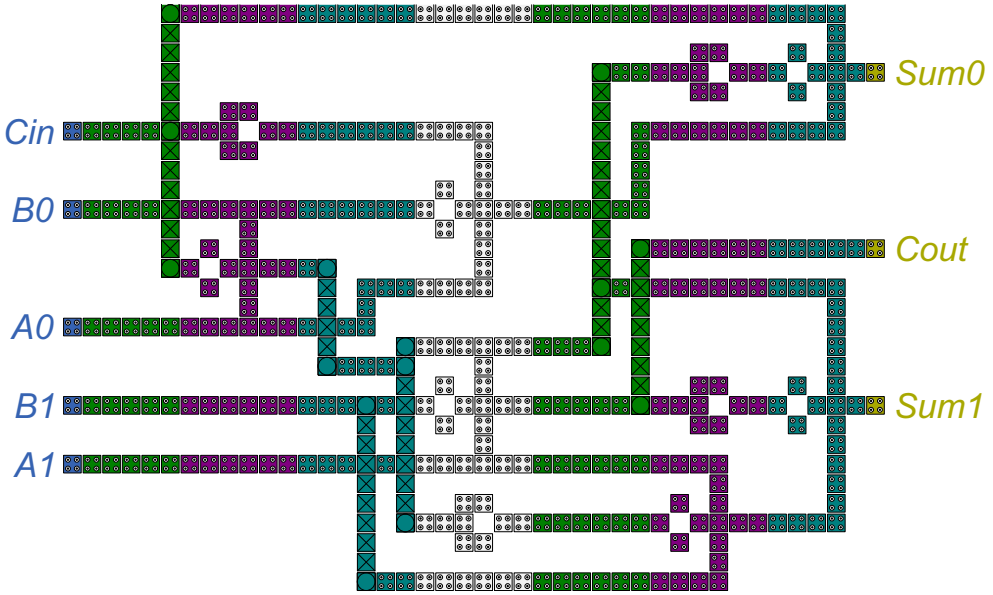
Slika 8.15 Začetna razmestitev T_{01} pri iskanju fizične razmestitve strukture dvobitnega polnega seštevalnika.



Slika 8.16 Rezultat prvega poskusa iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika pri začetni razmestitvi T_{01} je razmestitev T_{11}.



Slika 8.17 Rezultat drugega poskusa iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika pri začetni razmestitvi T_{01} je razmestitev T_{21}.



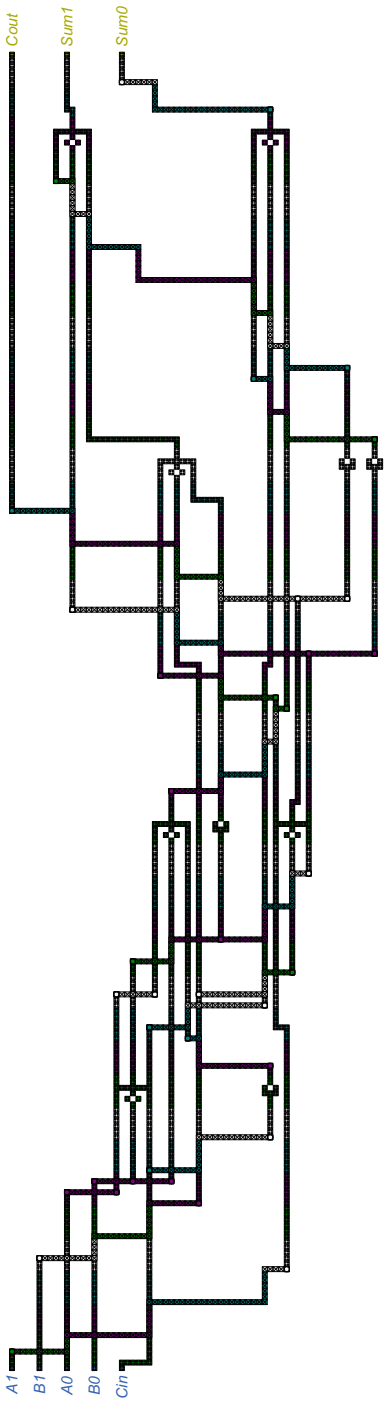
Slika 8.18 Rezultat tretjega poskusa iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika pri začetni razmestitvi T_{01} je razmestitev T_{31}.

- razmestitev T_{22}: $\sum_i N'_{dolzina_i} = 280$, $N'_{krizanje} = 10$, $N'_{kot} = 10$, cena$(T_{22}) = 380$, algoritem je našel razmestitev v iteraciji z indeksom 990 117;
- razmestitev T_{32}: $\sum_i N'_{dolzina_i} = 273$, $N'_{krizanje} = 9$, $N'_{kot} = 11$, cena$(T_{32}) = 373$, algoritem je našel razmestitev v iteraciji z indeksom 800 467.

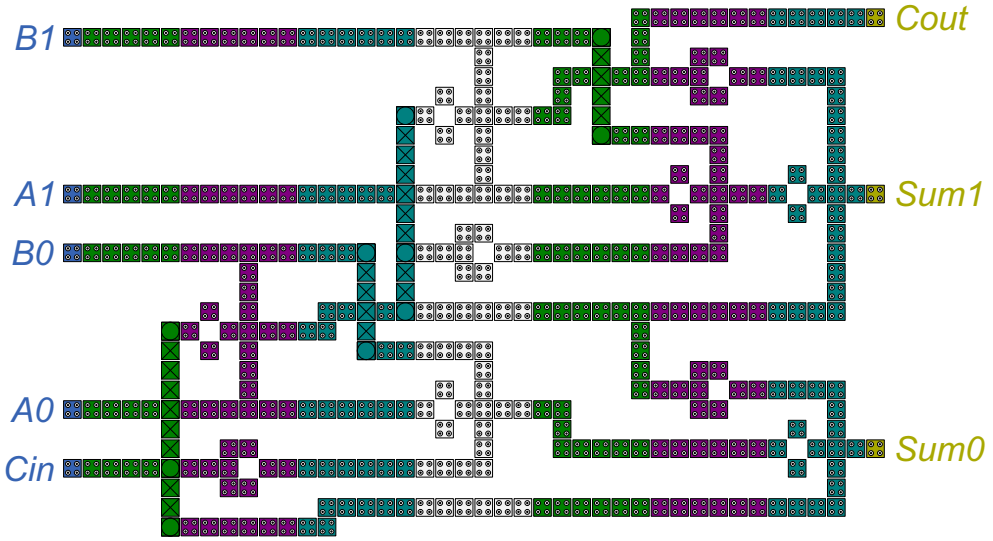
Pravilno delovanje avtomatsko zasnovanih struktur potrjuje grafični prikaz rezultatov simulacije, pridobljenih z uporabo verifikacijskega orodja QCADesigner, na sliki 8.23. Rezultati simulacij so enaki za vse fizične razmestitve struktur T_{11}, T_{21}, T_{31}, T_{12}, T_{22} in T_{32}.

Vrednosti optimizacijskih kriterijev naključno določenih začetnih razmestitev T_{01} in T_{02} so navedene v tabeli 8.4. Vrednosti optimiziranih kriterijev fizičnih razmestitev struktur T_{11}, T_{21}, T_{31}, T_{12}, T_{22} in T_{32} so navedene v tabeli 8.5.

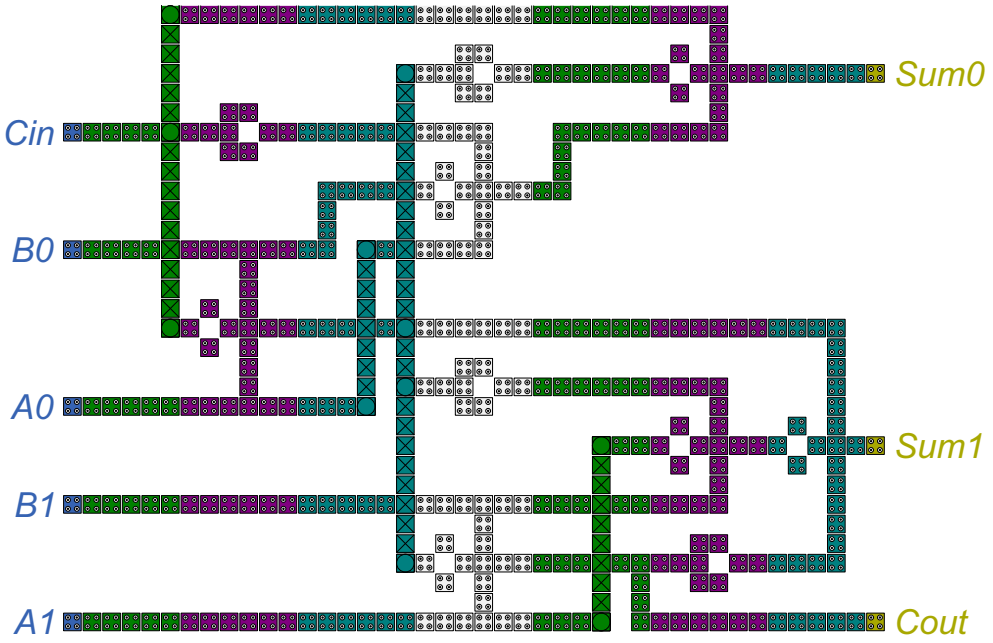
Pri neoptimiziranih začetnih razmestitvah T_{01} in T_{02} so dejanske vrednosti optimizacijskih kriterijev večje od ocenjenih vrednosti. Izračun cene razmestitve mora biti hiter, ker se izvede v vsaki iteraciji. Zato izračun cene podaja le oceno vrednosti kriterijev, pri čemer še niso znani dejanski položaji linij. Slednji so znani šele po končani fazi povezovanja. V neoptimiziranih razmestitvah so povezave dolge in med njimi obstaja veliko



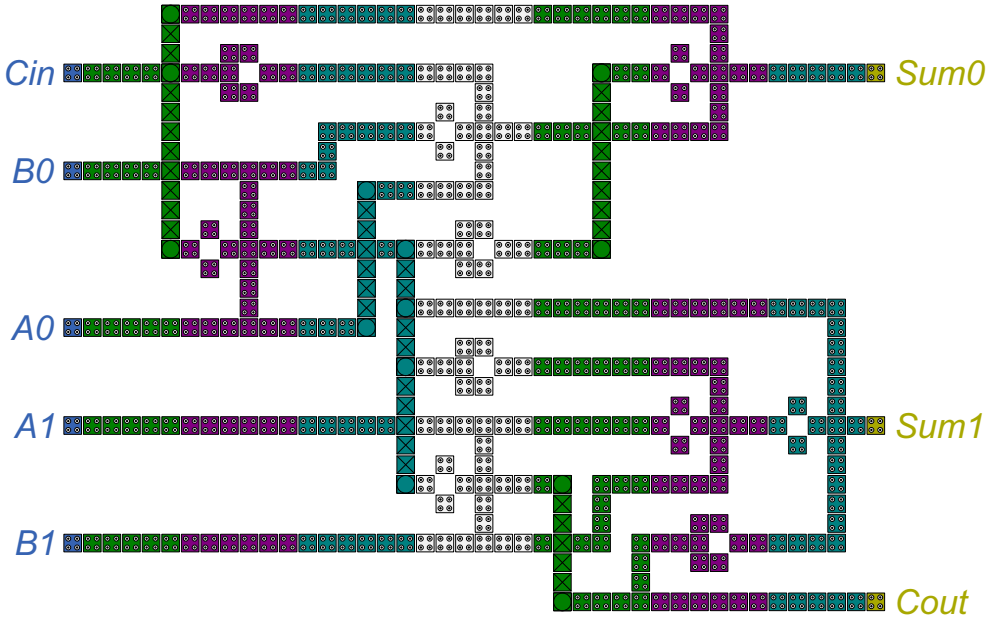
Slika 8.19 Začetna razmestitev T_{02} pri iskanju fizične razmestitve strukture dvobitnega polnega seštevalnika.



Slika 8.20 Rezultat prvega poskusa iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika pri začetni razmestitvi T_{02} je razmestitev T_{12}.



Slika 8.21 Rezultat drugega poskusa iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika pri začetni razmestitvi T_{02} je razmestitev T_{22}.

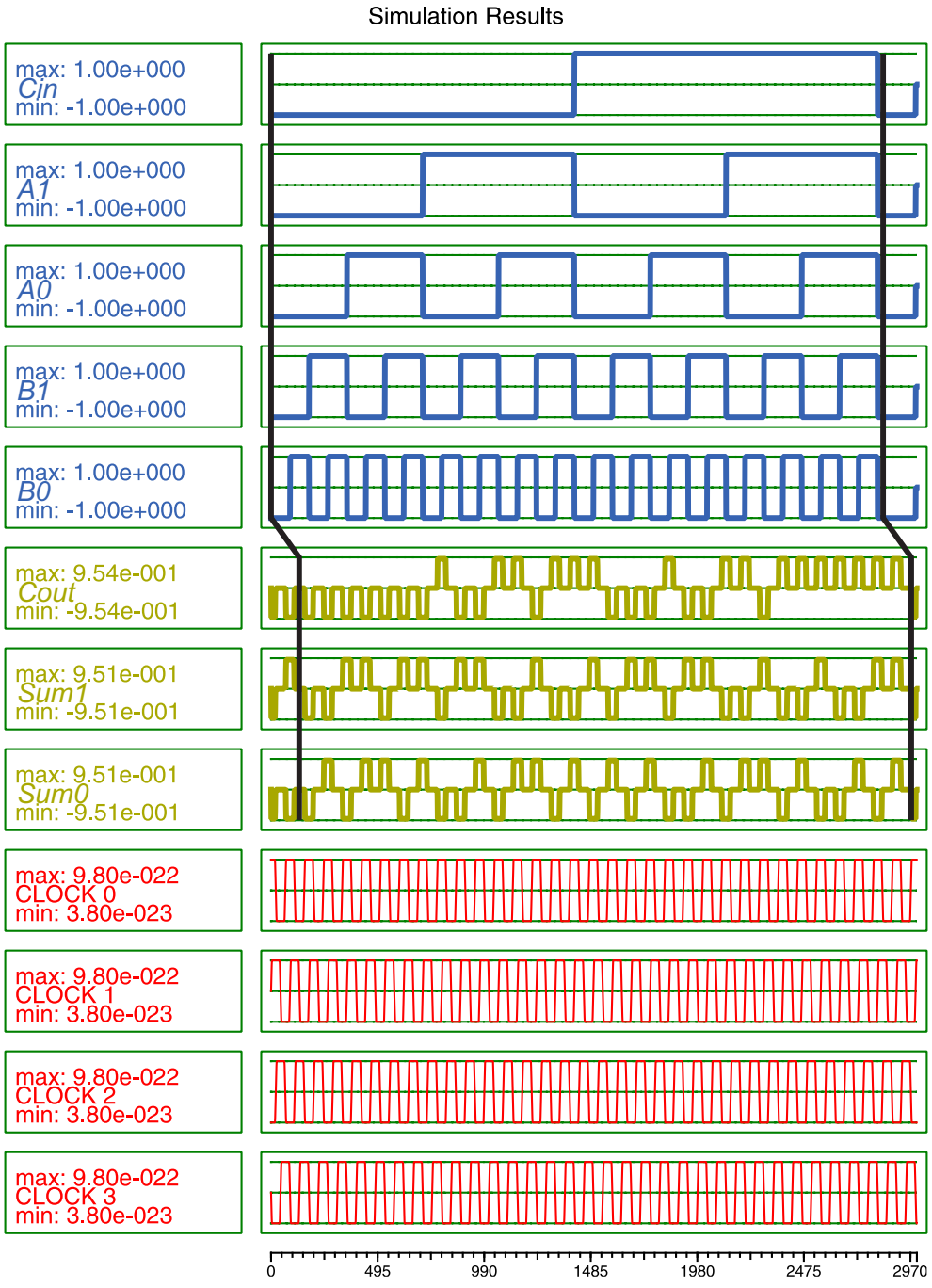


Slika 8.22 Rezultat tretjega poskusa iskanja fizične razmestitve strukture dvobitnega polnega seštevalnika pri začetni razmestitvi T_{02} je razmestitev T_{32}.

križanj. Zato je potrebno veliko število linij razdeliti med dodatne vmesne urine cone, tako da je zadoščeno načrtovalskim pravilom. Vsaka dodatna vmesna urina cona poveča število potrebnih urinih faz, porabljeno površino in vsoto dolžin vseh linij. Slednja se poveča za vsoto dolžin vseh linij, ki potekajo skozi dodatne vmesne cone. Pri razdelitvi navpičnih delov linij se poveča tudi število kotnih linij. Optimizacija kriterijev zato zelo izboljša kvaliteto fizične razmestitve strukture. Kot je razvidno iz rezultatov avtomatskega snovanja, so optimizirane avtomatsko zasnovane strukture precej kvalitetnejše od naključno določenih razmestitev.

Algoritem simuliranega ohlajanja združuje elemente naključnega in požrešnega iskanja. Na začetku njegovega izvajanja je iskanje podobno naključnemu, ker se skoraj vedno izvede prehod v novo stanje. V začetnem delu delovanja je namreč verjetnost za prehod v stanje z višjo ceno p_{visja_cena} blizu vrednosti 1, verjetnost prehoda v cenejše stanje pa je vselej enaka 1. Kasneje začne p_{visja_cena} padati proti 0, tako da so prehodi v dražja stanja vse redkejši. Na koncu se izvede še požrešna optimizacija najcenejšega do tedaj najdenega stanja.

Pri popolnoma naključnem iskanju je p_{visja_cena} enaka 1. V velikem številu iteracij



Slika 8.23 Grafični prikaz rezultatov simulacije delovanja fizične razmestitve strukture dvobitnega polnega seštevalnika. Zakasnitev signala v strukturi znaša 7 urinih faz oziroma 1,75 urinega cikla.

kriterij	T_{01}	T_{02}
število urinih faz	51	40
vsota dolžin vseh linij	3091	2165
število kotnih linij	134	105
porabljena površina	306×85	240×68
število križanj linij	55	52

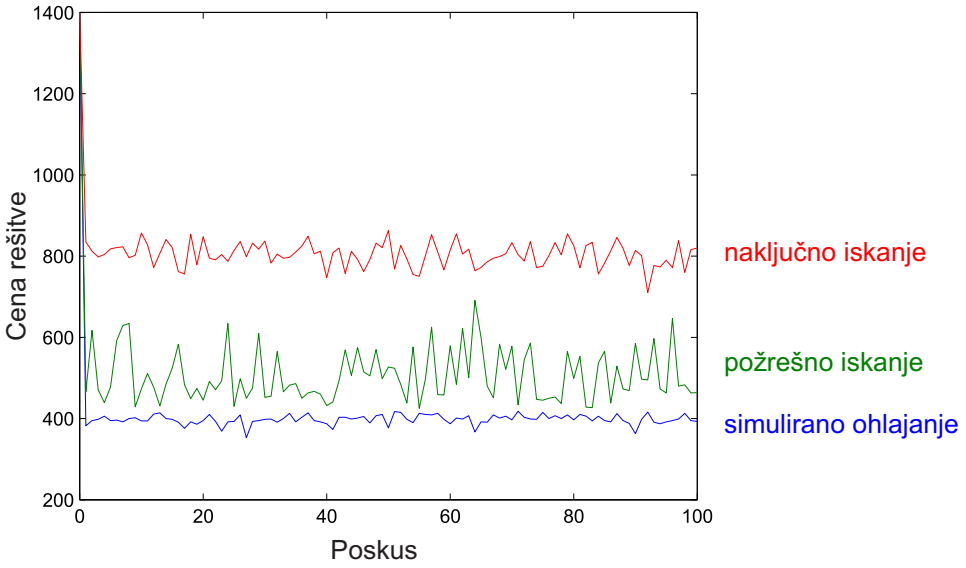
Tabela 8.4 Vrednosti optimizacijskih kriterijev naključno določenih začetnih razmestitev T_{01} in T_{02}.

kriterij	T_{11}	T_{21}	T_{31}	T_{12}	T_{22}	T_{32}
število urinih faz	7	7	7	7	7	7
vsota dolžin vseh linij	242	268	281	241	267	260
število kotnih linij	18	16	14	17	13	14
porabljena površina	42×27	42×29	42×30	42×27	42×32	42×31
število križanj linij	4	7	7	4	8	5

Tabela 8.5 Vrednosti optimiziranih kriterijev avtomatsko zasnovanih fizičnih razmestitev struktur T_{11}, T_{21}, T_{31}, T_{12}, T_{22} in T_{32}.

algoritem najde cenejšo rešitev od neoptimizirane naključno določene začetne razmestitve, vendar je takšna rešitev precej oddaljena od globalnega minimuma. Za požrešno iskanje v vsaki iteraciji velja $p_{visja\_cena} = 0$. Izvajajo se le prehodi v cenejša stanja, zato se požrešno iskanje ujame v lokalnem minimumu, ki je najbližje začetni razmestitvi. Algoritem simuliranega ohlajanja se lahko izogne takemu lokalnemu minimumu zaradi možnosti prehoda v stanje z višjo ceno. Tako lahko najde stanja, ki so cenejša od rezultata požrešnega iskanja.

Graf na sliki 8.24 prikazuje primerjavo rezultatov algoritmov simuliranega ohlajanja, požrešnega iskanja in naključnega iskanja. Izvedli smo 100 poskusov razmeščanja. V vsakem poskusu je bila začetna razmestitev T_{01} s ceno cena$(T_{01}) = 1400$. V poskusu z indeksom 0 razmeščanje ni bilo izvedeno, zato je rezultat kar začetna razmestitev T_{01}. Nato smo v vsakem poskusu ločeno izvedli razmeščanja z uporabo simuliranega ohlajanja, požrešnega iskanja in naključnega iskanja. Vsak algoritem je izvedel 1 000 000 iteracij v enem poskusu. Graf na sliki 8.24 prikazuje cene rešitev treh algoritmov iskanja v vsakem poskusu. Iz grafa je razvidno, da je v vsakem poskusu najcenejša rešitev rezultat



Slika 8.24 Cene rešitev algoritmov simuliranega ohlajanja, požrešnega iskanja in naključnega iskanja v stotih izvedenih poskusih. Začetna razmestitev je bila vselej T_{01}. V poskusu z indeksom 0 razmeščanje ni bilo izvedeno, zato je prikazana cena začetne razmestitve T_{01}. Najboljše rešitve je vselej dal algoritem simuliranega ohlajanja.

razmeščanja s simuliranim ohlajanjem, sledi rezultat požrešnega iskanja, najslabšo rešitev pa da naključno iskanje. Poleg tega lahko opazimo, da je najboljši rezultat izmed vseh rešitev požrešnega iskanja slabši od najslabšega rezultata simuliranega ohlajanja. Enaka primerjava velja med požrešnim in naključnim iskanjem.

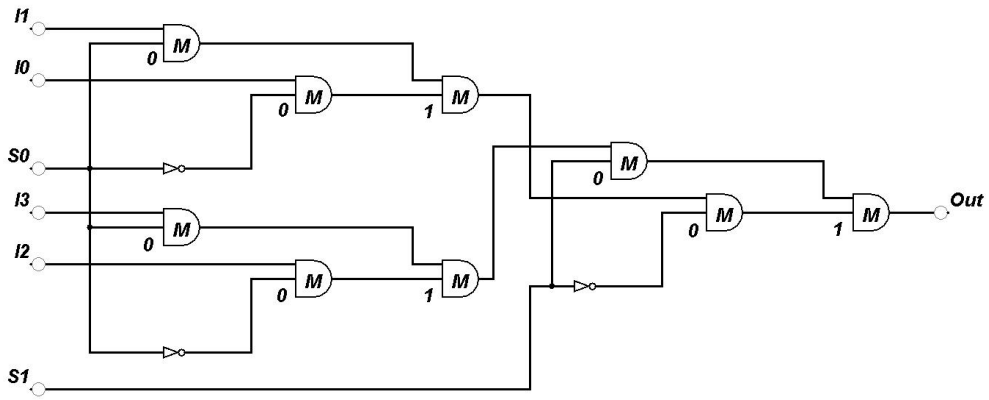
8.2.3 Struktura 4/1 multiplekserja

Struktura 4/1 multiplekserja na podlagi vrednosti dveh naslovnih vhodov $S0$ in $S1$ določi, kateri od podatkovnih vhodov $I0$, $I1$, $I2$, $I3$ se bo preslikal na izhod Out. Izhod je določen z enačbo

$$\begin{aligned} Out = \quad & I0 \cdot \overline{S1} \cdot \overline{S0} \ \vee \ I1 \cdot \overline{S1} \cdot S0 \ \vee \\ & I2 \cdot S1 \cdot \overline{S0} \ \vee \ I3 \cdot S1 \cdot S0. \end{aligned} \tag{8.6}$$

Logična shema 4/1 multiplekserja je prikazana na sliki 8.25.

Pri iskanju fizične razmestitve strukture 4/1 multiplekserja smo najprej določili začetno stanje razmestitve M_{01}, prikazano na sliki 8.26. Razmestitev je bila določena naključno, tako da je njena cena zelo verjetno veliko višja od cene optimalne razmestitve.



Slika 8.25 Logična shema 4/1 multiplekserja.

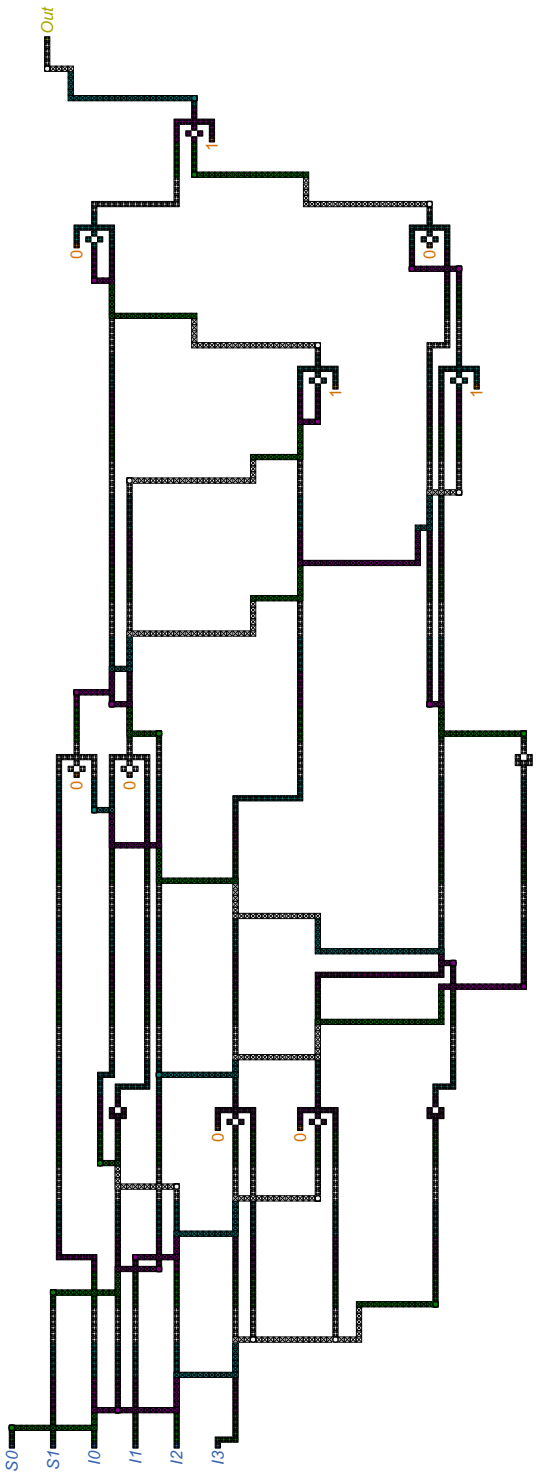
Za začetno razmestitev M_{01} je določena ocena vsote dolžin vseh linij $\sum_i N'_{dolzina_i} = 955$, ocena števila križanj linij $N'_{krizanje} = 35$ in ocena števila kotnih linij $N'_{kot} = 25$. Cena razmestitve se izračuna z uporabo enačbe (7.2) in je enaka cena$(M_{01}) = 1255$.

Nato smo izvedli tri poskuse avtomatskega iskanja, tako da je bila začetna razmestitev vedno M_{01}. V vsakem poskusu je bilo izvedenih 1 000 000 iteracij. Višina posamezne urine cone je znašala 100 področij. Rezultati poskusov so fizične razmestitve M_{11}, M_{21} in M_{31}, prikazane na slikah 8.27, 8.28 in 8.29.

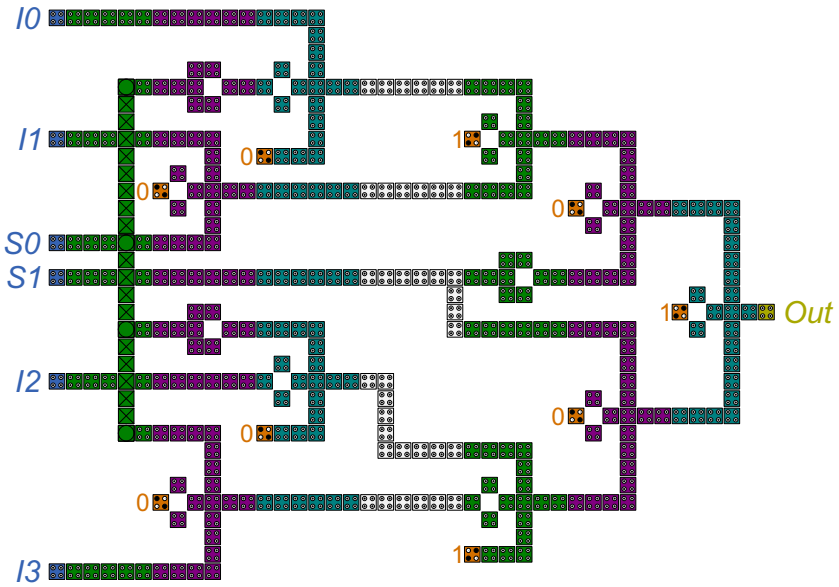
Za posamezno razmestitev veljajo ocene:

- razmestitev M_{11}: $\sum_i N'_{dolzina_i} = 164$, $N'_{krizanje} = 4$, $N'_{kot} = 5$, cena$(M_{11}) = 209$, algoritem je našel razmestitev v iteraciji z indeksom 775 861;
- razmestitev M_{21}: $\sum_i N'_{dolzina_i} = 171$, $N'_{krizanje} = 4$, $N'_{kot} = 5$, cena$(M_{21}) = 216$, algoritem je našel razmestitev v iteraciji z indeksom 779 039;
- razmestitev M_{31}: $\sum_i N'_{dolzina_i} = 167$, $N'_{krizanje} = 5$, $N'_{kot} = 6$, cena$(M_{31}) = 222$, algoritem je našel razmestitev v iteraciji z indeksom 766 720.

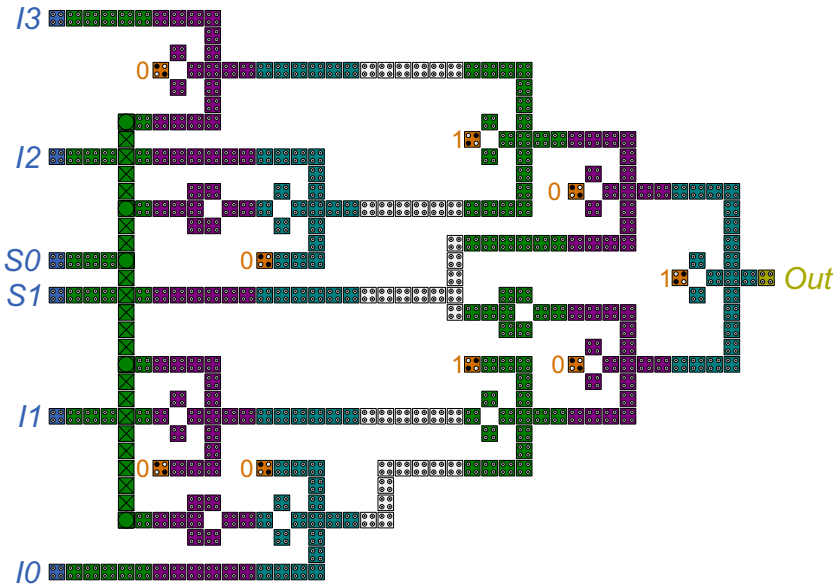
V drugem delu postopka iskanja fizične razmestitve strukture 4/1 multiplekserja smo določili novo začetno razmestitev M_{02}. Slednja se razlikuje od razmestitve M_{01}, obe pa sta bili določeni naključno. Razmestitev M_{02} je prikazana na sliki 8.30. Za začetno razmestitev M_{02} je določena ocena vsote dolžin vseh linij $\sum_i N'_{dolzina_i} = 810$, ocena števila križanj linij $N'_{krizanje} = 25$ in ocena števila kotnih linij $N'_{kot} = 22$. Cena razmestitve je enaka cena$(M_{02}) = 1045$.



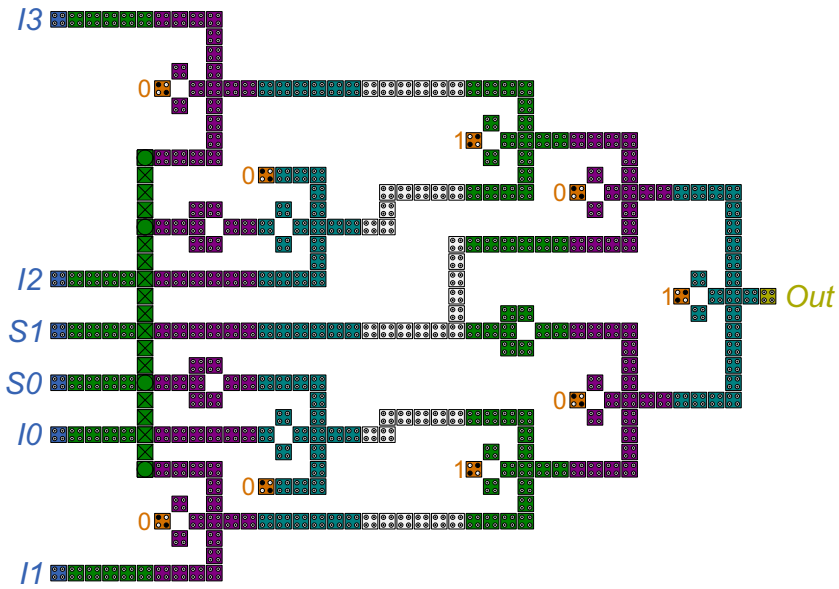
Slika 8.26 Začetna razmestitev M_{01} pri iskanju fizične razmestitve strukture 4/1 multiplekserja.



Slika 8.27 Rezultat prvega poskusa iskanja fizične razmestitve strukture 4/1 multiplekserja pri začetni razmestitvi M_{01} je razmestitev M_{11}.



Slika 8.28 Rezultat drugega poskusa iskanja fizične razmestitve strukture 4/1 multiplekserja pri začetni razmestitvi M_{01} je razmestitev M_{21}.



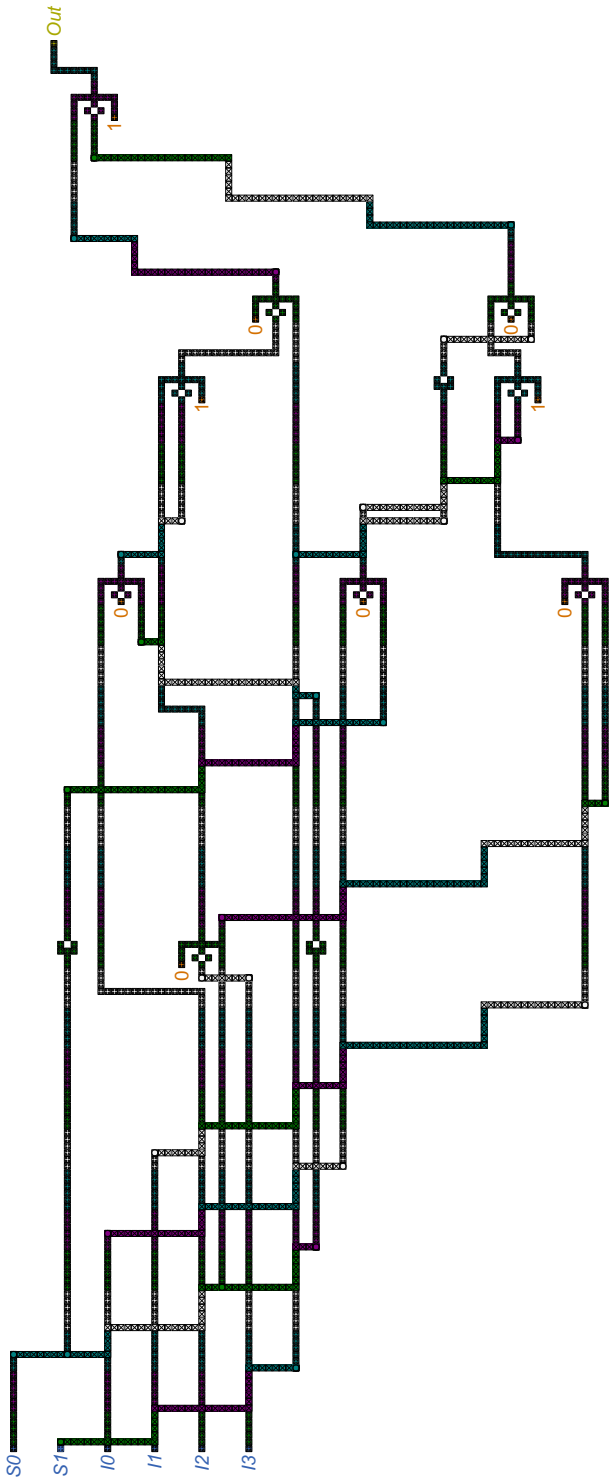
Slika 8.29 Rezultat tretjega poskusa iskanja fizične razmestitve strukture 4/1 multiplekserja pri začetni razmestitvi M_{01} je razmestitev M_{31}.

Nato smo ponovno izvedli tri poskuse avtomatskega iskanja. V vsakem poskusu je bilo izvedenih 1 000 000 iteracij. Višina posamezne urine cone je znašala 100 področij. Začetno stanje je vsakokrat določala razmestitev M_{02}. Rezultati poskusov so fizične razmestitve M_{12}, M_{22} in M_{32}, prikazane na slikah 8.31, 8.32 in 8.33.

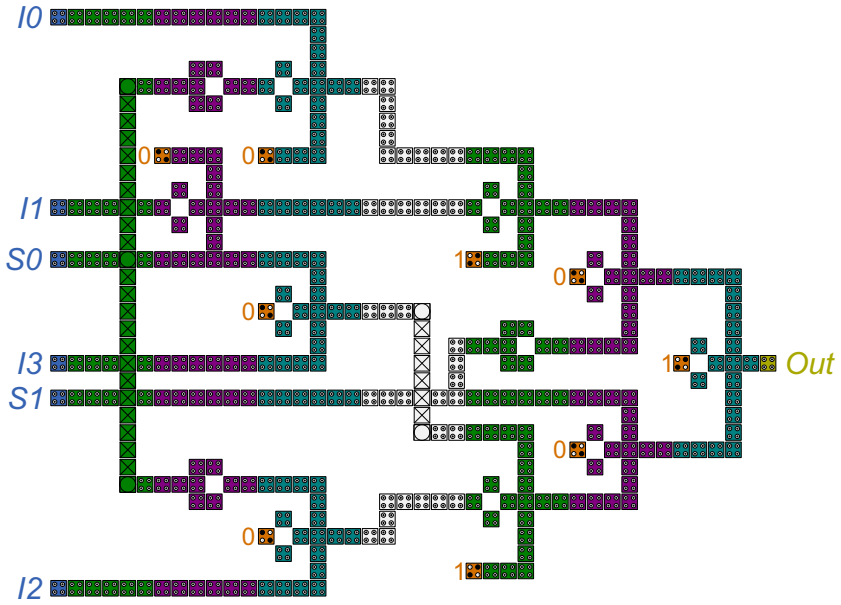
Za posamezno razmestitev veljajo ocene:

- razmestitev M_{12}: $\sum_i N'_{dolzina_i} = 165$, $N'_{krizanje} = 5$, $N'_{kot} = 6$, cena$(M_{12}) = 220$, algoritem je našel razmestitev v iteraciji z indeksom 772 249;
- razmestitev M_{22}: $\sum_i N'_{dolzina_i} = 179$, $N'_{krizanje} = 4$, $N'_{kot} = 6$, cena$(M_{22}) = 229$, algoritem je našel razmestitev v iteraciji z indeksom 818 565;
- razmestitev M_{32}: $\sum_i N'_{dolzina_i} = 176$, $N'_{krizanje} = 4$, $N'_{kot} = 7$, cena$(M_{32}) = 231$, algoritem je našel razmestitev v iteraciji z indeksom 824 115.

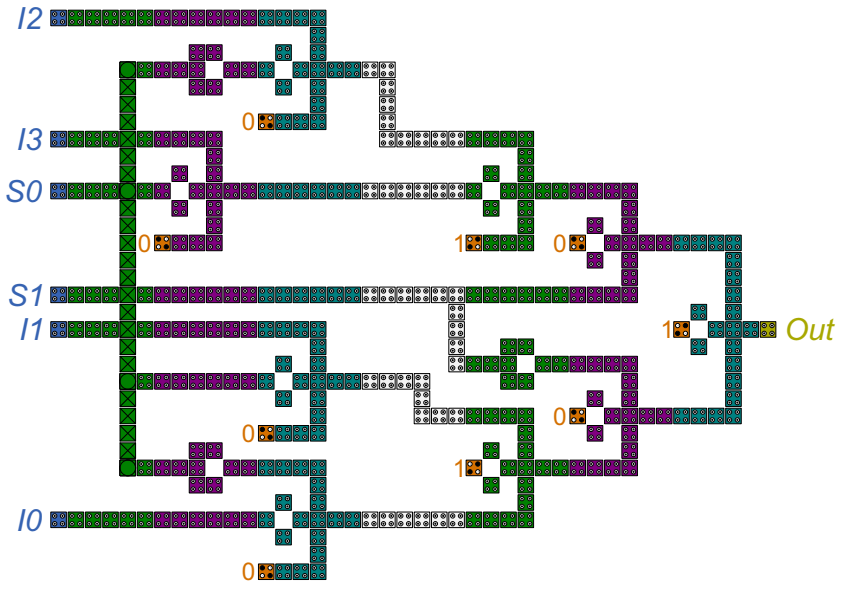
Pravilno delovanje avtomatsko zasnovanih struktur potrjuje grafični prikaz rezultatov simulacije, pridobljenih z uporabo verifikacijskega orodja QCADesigner, na sliki 8.34. Rezultati simulacij so enaki za vse fizične razmestitve struktur M_{11}, M_{21}, M_{31}, M_{12}, M_{22} in M_{32}.



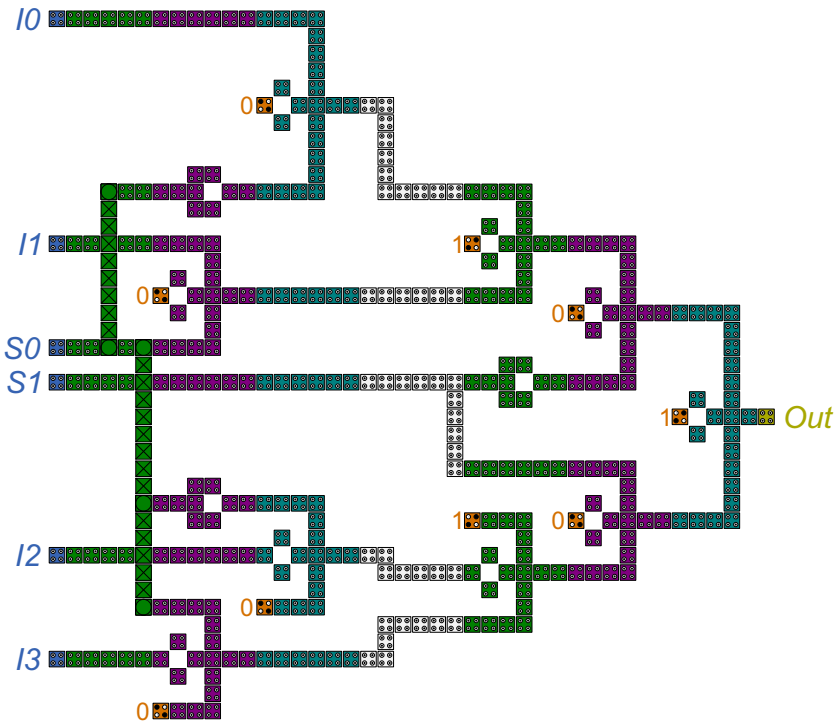
Slika 8.30 Začetna razmestitev M_{02} pri iskanju fizične razmestitve strukture 4/1 multiplekserja.



Slika 8.31 Rezultat prvega poskusa iskanja fizične razmestitve strukture 4/1 multiplekserja pri začetni razmestitvi M_{02} je razmestitev M_{12}.



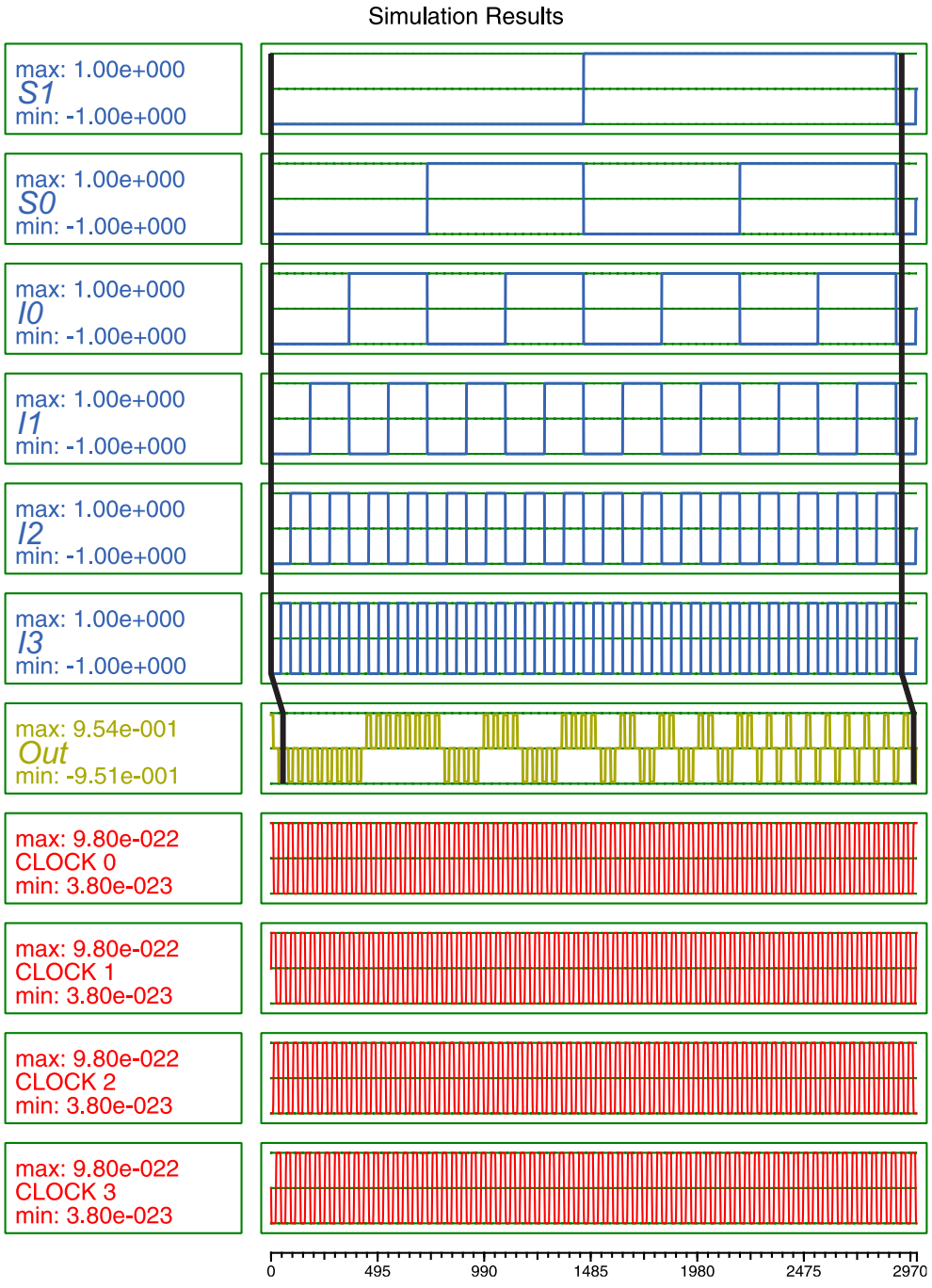
Slika 8.32 Rezultat drugega poskusa iskanja fizične razmestitve strukture 4/1 multiplekserja pri začetni razmestitvi M_{02} je razmestitev M_{22}.



Slika 8.33 Rezultat tretjega poskusa iskanja fizične razmestitve strukture 4/1 multiplekserja pri začetni razmestitvi M_{02} je razmestitev M_{32}.

Vrednosti optimizacijskih kriterijev naključno določenih začetnih razmestitev M_{01} in M_{02} so navedene v tabeli 8.6. Vrednosti optimiziranih kriterijev fizičnih razmestitev struktur M_{11}, M_{21}, M_{31}, M_{12}, M_{22} in M_{32} so navedene v tabeli 8.7.

Za primerjavo med avtomatsko zasnovanimi in že obstoječimi strukturami 4/1 multiplekserja so v tabeli 8.8 navedene vrednosti optimizacijskih kriterijev strukture, predstavljene v delu [93]. Multiplekser v delu [93] ne temelji na urinih conah s pravilnimi oblikami. Nekatere cone vsebujejo eno samo QCA celico, kar predstavlja veliko težavo za fizično realizacijo.



Slika 8.34 Grafični prikaz rezultatov simulacije delovanja fizične razmestitve strukture 4/1 multiplekserja. Zakasnitev signala v strukturi znaša 7 urinih faz oziroma 1,75 urinega cikla.

kriterij	M_{01}	M_{02}
število urinih faz	40	35
vsota dolžin vseh linij	2028	1798
število kotnih linij	103	89
porabljena površina	240×89	210×89
število križanj linij	32	38

Tabela 8.6 Vrednosti optimizacijskih kriterijev naključno določenih začetnih razmestitev M_{01} in M_{02}.

kriterij	M_{11}	M_{21}	M_{31}	M_{12}	M_{22}	M_{32}
število urinih faz	7	7	7	7	7	7
vsota dolžin vseh linij	144	150	140	158	156	156
število kotnih linij	5	6	7	9	7	9
porabljena površina	42×33	42×33	42×33	42×34	42×33	42×41
število križanj linij	3	3	3	4	3	3

Tabela 8.7 Vrednosti optimiziranih kriterijev avtomatsko zasnovanih fizičnih razmestitev struktur M_{11}, M_{21}, M_{31}, M_{12}, M_{22} in M_{32}.

kriterij	[93]
število urinih faz	6
vsota dolžin vseh linij	149
število kotnih linij	8
porabljena površina	20×31
število križanj linij	8

Tabela 8.8 Vrednosti optimizacijskih kriterijev fizične razmestitve strukture 4/1 multiplekserja, predstavljene v delu [93].

8.3 Procesna zahtevnost iskanja rešitve

Aplikacija za avtomatsko snovanje fizičnih razmestitev struktur QCA se je izvajala na dvojedrnem procesorju Intel Core 2 Duo s frekvenco delovanja 2,66 GHz. V vsakem poskusu snovanja strukture smo izmerili čas, ki je potekel od zagona aplikacije do trenutka, ko je bila izdelana izhodna datoteka z zasnovano strukturo v QCADesigner formatu. Časi, merjeni v sekundah, so zbrani v tabeli 8.9.

	enobitni seštevalnik	dvobitni seštevalnik	4/1 multiplekser
poskus 1	8,767 s	24,382 s	24,351 s
poskus 2	9,016 s	24,242 s	24,772 s
poskus 3	9,094 s	24,102 s	24,445 s
poskus 4	9,079 s	23,790 s	24,336 s
poskus 5	9,032 s	23,602 s	24,382 s
poskus 6	9,048 s	23,712 s	24,243 s
povprečni čas	9,006 s	23,972 s	24,422 s

Tabela 8.9 Časi izvajanja aplikacije avtomatskega snovanja struktur, merjeni v sekundah. V poskusu 1 so bile zasnovane strukture S_{11}, T_{11}, M_{11}, v poskusu 2 strukture S_{21}, T_{21}, M_{21}, v poskusu 3 strukture S_{31}, T_{31}, M_{31}, v poskusu 4 strukture S_{12}, T_{12}, M_{12}, v poskusu 5 strukture S_{22}, T_{22}, M_{22} in v poskusu 6 strukture S_{32}, T_{32}, M_{32}.

8.4 Primerjava razvitih metod z že obstoječimi metodami snovanja QCA

Ročno razvite razmestitve niso zasnovane s formalnim postopkom, zato slednjega ni mogoče posplošiti na snovanje drugih struktur in ga avtomatizirati. Obstoječe ročno razvite razmestitve večinoma ne upoštevajo značilnosti kvantnih celičnih avtomatov, zato imajo manjšo možnost tehnološke izvedbe. Nekatere razmestitve so zasnovane na urinih conah nepravilnih oblik, ki vsebujejo majhno število celic. To sicer zmanjša število vseh celic v strukturi in porabljeno površino, vendar bi bilo tako strukturo izredno težko izdelati [2].

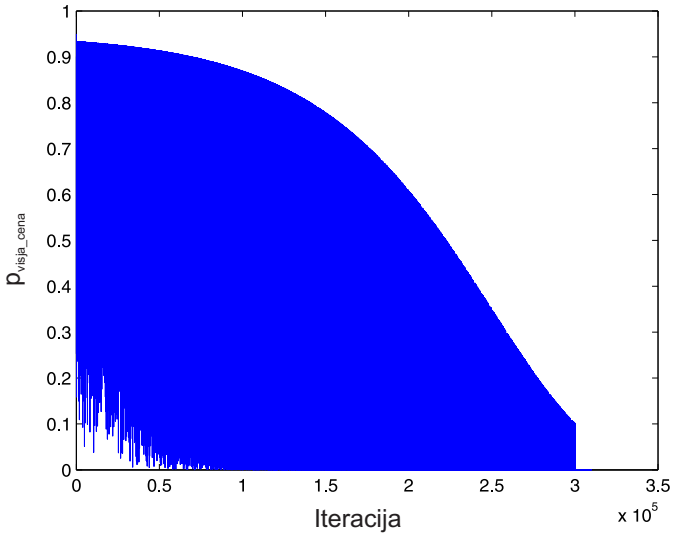
Avtorji [86, 32, 23, 88, 24] so se ukvarjali z odpravo vseh križanj linij v QCA strukturi z uporabo podvojevanja logičnih primitivov in vhodnih priključkov. Ta metoda minimizira oziroma odpravi vsa križanja v strukturi na račun eksponentnega povečanja števila uporabljenih logičnih primitivov in vhodnih priključkov. Pri tem je potrebno na podvo-

jene vhodne priključke pripeljati isti signal, kar v praksi pomeni, da se vsa križanja linij prestavijo pred vhodne priključke obravnavane QCA strukture. V primeru, da se slednja uporablja kot modul znotraj večjega sistema, je torej še vedno potrebno poiskati rešitev za realizacijo križanja linij.

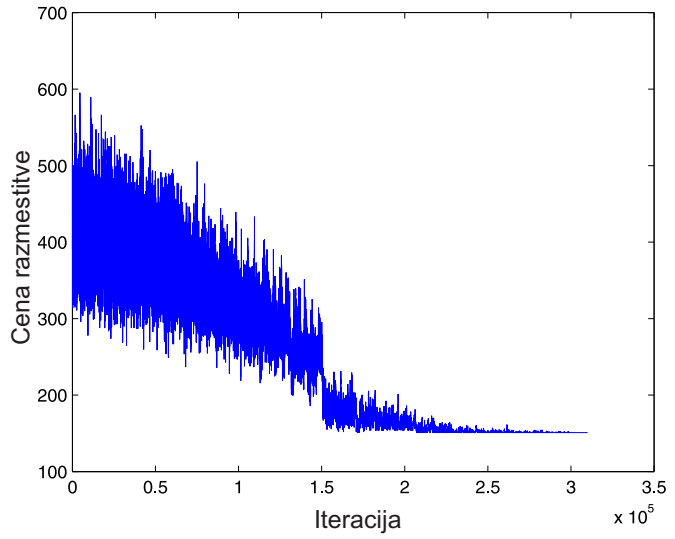
8.5 Možnosti za izboljšave iskanja rešitve

Iz grafov na slikah 8.6(a), 8.7(a) in 8.8(a) je razvidno, da je verjetnost prehoda v stanje z višjo ceno p_{visja_cena} na začetku delovanja algoritma precej časa blizu vrednosti 0,95. Vrednost p_{visja_cena} se začne bistveno spreminjati šele po iteraciji z indeksom 400 000. Zaradi velikega števila prehodov v stanja z višjo ceno na začetku delovanja algoritma cena razmestitve zelo niha. Konvergirati začne šele po iteraciji z indeksom 600 000. V zadnjem delu delovanja algoritma, od iteracije z indeksom 900 000 do iteracije z indeksom 990 000, se cena ne spreminja več. Takrat se razmestitev nahaja blizu ali v lokalnem minimumu $lmin$, zaradi nizke vrednosti p_{visja_cena} pa se prehodi v stanja z višjo ceno ne pojavljajo več. Cena razmestitve pade kvečjemu v zadnjem procentu iteracij, ko se izvaja požrešna optimizacija. Cena lokalnega minimuma $lmin$ ni veliko višja od cene globalnega minimuma, ali pa sta v najboljšem primeru ceni celo enaki. Zaradi navedenih dejstev ugotavljamo, da je izvajanje iteracij med 1 in 600 000 ter iteracij med 900 000 in 990 000 nepotrebno. Izvedli smo poskus, pri katerem se v navedenih iteracijah niso izračunavala nova stanja, v ostalih iteracijah pa je postopek ostal nespremenjen. Po pričakovanjih smo dobili podobne rezultate kot pri nespremenjenem postopku, v katerem se novo stanje izračuna v vsaki iteraciji. Na sliki 8.35(a) je prikazan graf verjetnosti prehoda v stanje z višjo ceno v odvisnosti od indeksa iteracije. Podatki so bili zbrani v poskusu, v katerem se je novo stanje izračunavalo le v iteracijah med 600 000 in 900 000 ter v požrešni optimizaciji v zadnjih 10 000 iteracijah. Slika 8.35(b) prikazuje graf odvisnosti cene razmestitve glede na indeks iteracije. Na začetku delovanja algoritma cena razmestitve lokalno precej niha, vendar globalno pada proti ceni končne rešitve.

Koeficient k_6 določa utež za razliko med cenama trenutnega stanja s_{i-1} in novega stanja s_i, kadar je cena novega stanja višja od cene trenutnega stanja. Razlika med cenama je enaka Δc=cena(s_i)-cena(s_{i-1}). Pri veliki vrednosti koeficienta k_6 ima Δc velik vpliv na izračun verjetnosti $p(s_i|s_{i-1})$ v enačbi (7.5), pri majhni vrednosti k_6 pa je vpliv Δc na izračun $p(s_i|s_{i-1})$ majhen. Za preučitev vpliva vrednosti k_6 na iskanje



(a)



(b)

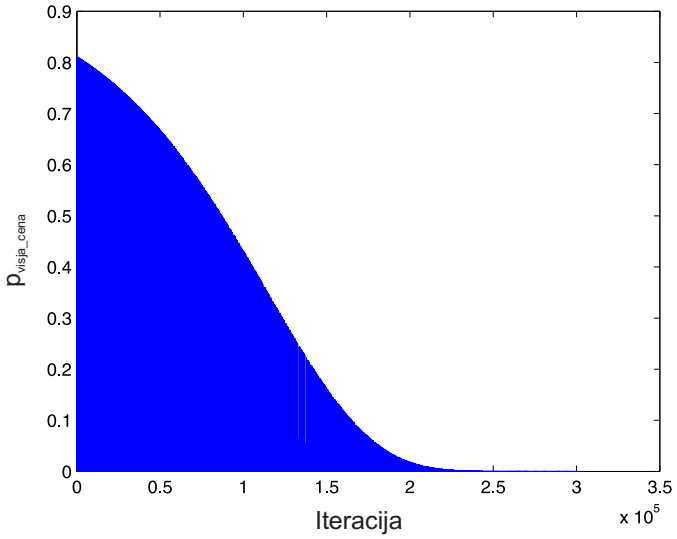
Slika 8.35 Graf odvisnosti verjetnosti prehoda v stanje z višjo ceno glede na indeks iteracije (a). Graf odvisnosti cene razmestitve glede na indeks iteracije (b). Podatki so bili zbrani v poskusu, v katerem se je novo stanje izračunavalo v iteracijah med 600 000 in 900 000 ter v požrešni optimizaciji v zadnjih 10 000 iteracijah. Tako je bilo izvedenih 310 000 iteracij.

rešitve smo izvedli dva poskusa, prvega z veliko in drugega z majhno vrednostjo k_6.

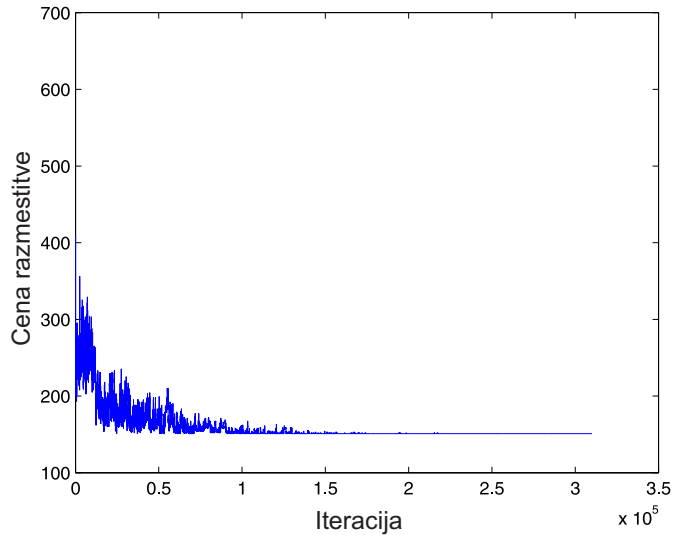
V prvem poskusu smo določili vrednost k_6=10. Verjetnosti prehoda v stanje z višjo ceno in cene razmestitev v 310 000 izvedenih iteracijah so prikazane na sliki 8.36. V tem poskusu je imel Δc velik vpliv na izračun $p(s_i|s_{i-1})$. Čim večja je bila vrednost Δc, tem manjša je bila verjetnost $p(s_i|s_{i-1})$. Zato se je v postopku iskanja pogosteje izvedel prehod v novo stanje s_i, pri katerem je bila vrednost Δc majhna. Kot je razvidno iz grafa na sliki 8.36(b), je cena razmestitve hitro padla in se nato ustalila pri vrednosti cene končne rešitve. Slednja se je nahajala v lokalnem minimumu, ki je ležal blizu začetne razmestitve. Ker je iskanje obiskalo manjše število lokalnih minimumov, končna rešitev pogosto ni bila tako kvalitetna kot v primeru uporabe vrednosti k_6, določene z izrazom (7.7).

V drugem poskusu smo izbrali majhno vrednost k_6=0,1. Verjetnosti prehoda v stanje z višjo ceno in cene razmestitev v 310 000 izvedenih iteracijah so prikazane na sliki 8.37. Vpliv Δc na izračun $p(s_i|s_{i-1})$ je bil majhen. Verjetnost prehoda v stanje z višjo ceno je padala zelo počasi, kot prikazuje graf na sliki 8.37(a). Ker je bilo izvedenih veliko prehodov v stanja z višjo ceno, je vrednost cene razmestitve zelo nihala. Šele proti koncu delovanja algoritma se je približala ceni končne rešitve. Ker se verjetnost prehoda v stanje z višjo ceno na koncu delovanja algoritma ni približala 0, iskanje pogosto ni našlo kvalitetnega lokalnega minimuma s ceno, ki ni veliko višja od cene globalnega minimuma. Končna razmestitev je bila večkrat ocenjena slabše kot razmestitev, ki jo je našel algoritem z uporabo vrednosti k_6, določene z izrazom (7.7).

Izvedli smo tudi poskus, pri katerem se je optimiziralo več razmestitev hkrati. V začetnih 250 000 iteracijah smo izbrali 5 različnih razmestitev in jih optimizirali s požrešnim iskanjem. Prvih 250 000 iteracij smo razdelili na 5 enako dolgih intervalov I_1, I_2, I_3, I_4 in I_5. Vsak interval je tako vseboval 50 000 iteracij. Interval I_1 je vseboval iteracije med 1 in 50 000, I_2 je vseboval iteracije med 50 001 in 100 000, I_3 je vseboval iteracije med 100 001 in 150 000, I_4 je vseboval iteracije med 150 001 in 200 000 ter I_5 je vseboval iteracije med 200 001 in 250 000. Najbolje ocenjeno razmestitev na intervalu I_i smo označili z M_i. Tako smo dobili 5 razmestitev M_1, M_2, M_3, M_4 in M_5. Vsako od teh razmestitev smo optimizirali s požrešnim iskanjem v 10 000 iteracijah. Običajno je imela najnižjo ceno optimizacija razmestitve M_5. Interval I_5 vsebuje iteracije, ki se izvedejo v zadnjem delu iskanja. Zato se razmestitve, ki so izračunane v iteracijah znotraj I_5, pogosto nahajajo blizu lokalnega minimuma z nizko ceno. To pomeni, da je najkoristneje

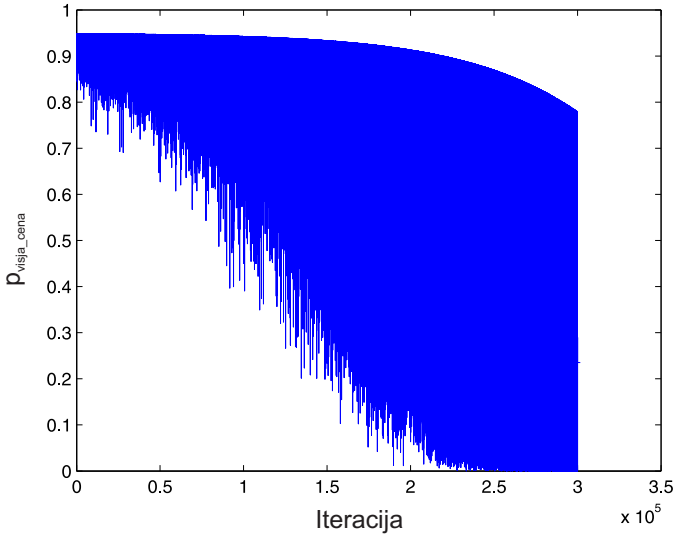


(a)

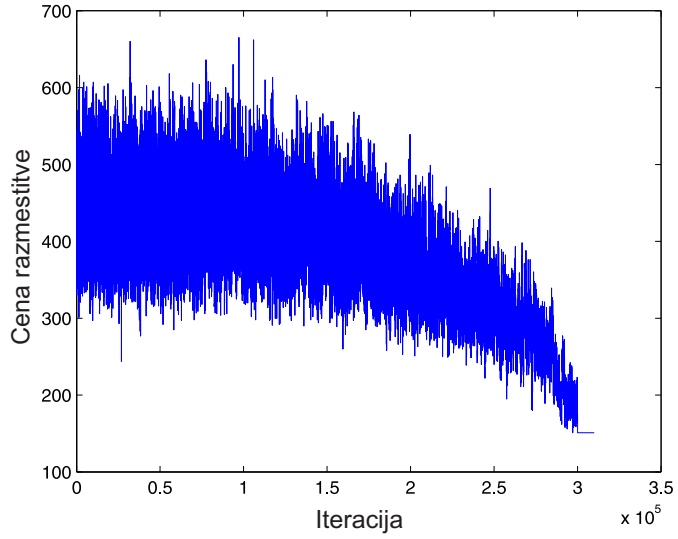


(b)

Slika 8.36 Graf odvisnosti verjetnosti prehoda v stanje z višjo ceno glede na indeks iteracije (a). Graf odvisnosti cene razmestitve glede na indeks iteracije (b). V poskusu je bilo izvedenih 310 000 iteracij. Vrednost koeficienta k_6 je bila 10.



(a)



(b)

Slika 8.37 Graf odvisnosti verjetnosti prehoda v stanje z višjo ceno glede na indeks iteracije (a). Graf odvisnosti cene razmestitve glede na indeks iteracije (b). V poskusu je bilo izvedenih 310 000 iteracij. Vrednost koeficienta k_6 je bila 0,1.

optimizirati razmestitve, dobljene v zadnjem delu izvajanja algoritma.

9 Opis razvite aplikacije

9.1 Izbira programskega jezika in razvojnega okolja

Aplikacija je napisana v programskem jeziku C++. Ta programski jezik je dobro poznan in pogosto uporabljan pri implementaciji aplikacij. Za podporo razvoju programske opreme v jeziku C++ je na voljo veliko uporabnih knjižnic. Tako je omogočeno učinkovito delo z uporabo dobro preizkušenih standardnih knjižnic. C++ omogoča programiranje na nizkem in visokem nivoju. Je večnamenski programski jezik in je uporaben za razvoj raznovrstnih tipov aplikacij. Prevajalnik jezika C++ prevede izvorno kodo v strojni jezik, ki je prirejen za procesorsko arhitekturo, na kateri se bo izvajala aplikacija. Takšna strojna koda omogoča hitro izvajanje programa. Zaradi dostopnosti številnih uporabnih knjižnic in hitrosti smo za razvojni programski jezik izbrali C++.

Za razvojno okolje smo uporabili Microsoft Visual Studio 2010. To okolje med drugim nudi urejevalnik programske kode in razhroščevalnik. Razvojno okolje močno poenostavi razvoj obsežne aplikacije. Okolje Microsoft Visual Studio 2010 smo izbrali zato, ker omogoča dobro podporo programiranju v jeziku C++.

9.2 Shema aplikacije

Aplikacijo "place and route" sestavlja šest projektov, ki so našteti v naslednjih alinejah:

- QCA entities,
- parse QCA design,
- floorplanner,
- placer,
- router in
- place and route.

Projekt "QCA entities" vsebuje razrede, ki definirajo osnovne entitete v QCA, kot so urina cona, priključek, nadgrajena majoritetna vrata, negator in povezava.

V projektu "parse QCA design" so implementirane metode, ki preberejo vhodne podatke in na njihovi podlagi ustvarijo ustrezne podatkovne strukture. Slednje nato uporabljajo metode za snovanje fizične razmestitve strukture QCA. Projekt vsebuje tudi metode za izdelavo izhodne datoteke v QCADesigner formatu.

Metode v projektu "floorplanner" določijo geometrijo strukture. Pri tem se določi seznam potrebnih urinih con za realizacijo strukture. V seznam se lahko kasneje po potrebi doda vmesne cone. Za vsak logični primitiv se določijo urine cone, v katerih je lahko nameščen (razdelek 7.2).

Projekt "placer" vsebuje metode za razmeščanje vhodno/izhodnih priključkov in logičnih primitivov. Metoda "simulated_annealing()" izvede razmeščanje z uporabo simuliranega ohlajanja, ki je opisano v razdelku 7.3.

V projektu "router" so implementirane metode za povezovanje vhodno/izhodnih priključkov in logičnih primitivov. Metoda "multilayer_router()" določi položaje vseh linij v povezavah, kot je opisano v razdelku 7.4.

Projekt "place and route" vsebuje metodo "main()", ki zažene aplikacijo. Ob zagonu se najprej preberejo vhodni podatki. Nato se zaporedoma izvede snovanje geometrije strukture, razmeščanje in nazadnje povezovanje. Ko je rešitev določena, se rezultat zapiše v izhodno datoteko.

9.3 Opis vhodnih in izhodnih podatkov

Vhodne podatke vsebuje datoteka v formatu XML. V njej so našteti vhodno/izhodni priključki strukture, logični primitivi in povezave med njimi. Spodaj je navedena vsebina vhodne datoteke s podatki za snovanje fizične razmestitve strukture enobitnega seštevalnika na podlagi logične sheme s slike 8.1.

```
<?xml version="1.0" encoding='utf-8'?>
<!--enobitni polni seštevalnik-->
<!--vsebuje 3 majoritetna vrata in 2 negatorja-->
<!--vhodi A, B, Cin; izhoda Cout, Sum-->
<full_adder_1_bit>

  <!--višina urine cone-->
  <grid_height value='50'/>

  <!--vhodno/izhodni priključki strukture-->
  <cells>
    <!--vhodni priključki strukture-->
    <!--so izhodni priključki povezav znotraj strukture-->
    <output_cell name='A'>
      <net>
        <out_pin QCA_gate='m1' pin='1'/>
        <out_pin QCA_gate='m2' pin='1'/>
      </net>
    </output_cell>
    <output_cell name='B'>
      <net>
        <out_pin QCA_gate='m1' pin='2'/>
        <out_pin QCA_gate='m2' pin='2'/>
      </net>
    </output_cell>
    <output_cell name='Cin'>
      <net>
```

```
        <out_pin QCA_gate='m1' pin='3'/>
        <out_pin QCA_gate='i1' pin='1'/>
        <out_pin QCA_gate='m3' pin='2'/>
      </net>
    </output_cell>
    <!--izhodni priključki strukture-->
    <!--so vhodni priključki povezav znotraj strukture-->
    <input_cell name='Cout'>
    </input_cell>
    <input_cell name='Sum'>
    </input_cell>
  </cells>

  <!--logični primitivi-->
  <QCA_gates>
    <majority_gate name='m1'>
      <net>
        <out_pin QCA_gate='i2' pin='1'/>
        <out_pin pin='Cout'/>
      </net>
    </majority_gate>
    <majority_gate name='m2'>
      <net>
        <out_pin QCA_gate='m3' pin='3'/>
      </net>
    </majority_gate>
    <majority_gate name='m3'>
      <net>
        <out_pin pin='Sum'/>
      </net>
    </majority_gate>
    <inverter_gate name='i1'>
      <net>
```

```
        <out_pin QCA_gate='m2' pin='3'/>
      </net>
    </inverter_gate>
    <inverter_gate name='i2'>
      <net>
        <out_pin QCA_gate='m3' pin='1'/>
      </net>
    </inverter_gate>
  </QCA_gates>

</full_adder_1_bit>
```

Vhodna datoteka vsebuje strukturirane podatke, določene na podlagi logične sheme. V segmentu z etiketo "cells" so našteti vhodni in izhodni priključki strukture. Vhodni priključki strukture, ki služijo kot izhodni priključki povezav znotraj strukture, so podani kot elementi z etiketo "output_cell". Vsak vhodni priključek strukture ima podano ime in segment z etiketo "net". V slednjem so našteti vsi priključki, s katerimi je povezan izbrani vhodni priključek strukture. Priključki v tem segmentu so podani kot elementi z etiketo "out_pin". Če gre za izhodni priključek strukture, je podano njegovo ime, sicer pa sta navedena ime logičnega primitiva in pozicija na primitivu, na kateri se priključek nahaja. Ime logičnega primitiva je določeno z atributom "QCA_gate", pozicija pa z atributom "pin". Pozicija 1 določa zgornji priključek nadgrajenih majoritetnih vrat, pozicija 2 srednji priključek in pozicija 3 spodnji priključek.

V segmentu z etiketo "cells" so izhodni priključki strukture, ki služijo kot vhodni priključki povezav znotraj strukture, podani kot elementi z etiketo "input_cell". Ime priključka določa atribut "name".

Segmentu "cells" sledi segment z etiketo "QCA_gates", v katerem so našteti logični primitivi. Nadgrajena majoritetna vrata so podana kot elementi z etiketo "majority_gate", negatorji pa kot elementi z etiketo "inverter_gate". Vsak logični primitiv ima podano ime z atributom "name" in vsebuje segment z etiketo "net". V tem segmentu so našteti vsi priključki, s katerimi je povezan izhodni priključek izbranega logičnega primitiva. Segment "net" pri logičnih primitivih je strukturiran na enak način kot segment "net" pri vhodnih priključkih strukture.

Izhodni podatki so zapisani v datoteki v QCADesigner formatu. Izhodna datoteka vsebuje celotno zasnovano fizično razmestitev strukture z natanko določenimi položaji QCA celic. Zasnovano strukturo je možno enostavno verificirati z uporabo simulacije v orodju QCADesigner. Primer datoteke v QCADesigner formatu z opisom ene same QCA celice je v tekstovni obliki podan spodaj.

```
[VERSION]
qcadesigner_version=2.000000
[#VERSION]
[TYPE:DESIGN]
[TYPE:QCADLayer]
type=3
status=1
pszDescription=Drawing Layer
[#TYPE:QCADLayer]
[TYPE:QCADLayer]
type=0
status=2
pszDescription=Substrate
[TYPE:QCADSubstrate]
[TYPE:QCADStretchyObject]
[TYPE:QCADDesignObject]
x=3000.000000
y=1500.000000
bSelected=FALSE
clr.red=65535
clr.green=65535
clr.blue=65535
bounding_box.xWorld=0.000000
bounding_box.yWorld=0.000000
bounding_box.cxWorld=6000.000000
bounding_box.cyWorld=3000.000000
[#TYPE:QCADDesignObject]
```

```
[#TYPE:QCADStretchyObject]
grid_spacing=20.000000
[#TYPE:QCADSubstrate]
[#TYPE:QCADLayer]
[TYPE:QCADLayer]
type=1
status=0
pszDescription=Main Cell Layer
[TYPE:QCADCell]
[TYPE:QCADDesignObject]
x=60.000000
y=60.000000
bSelected=FALSE
clr.red=0
clr.green=65535
clr.blue=0
bounding_box.xWorld=51.000000
bounding_box.yWorld=51.000000
bounding_box.cxWorld=18.000000
bounding_box.cyWorld=18.000000
[#TYPE:QCADDesignObject]
cell_options.cxCell=18.000000
cell_options.cyCell=18.000000
cell_options.dot_diameter=5.000000
cell_options.clock=0
cell_options.mode=QCAD_CELL_MODE_NORMAL
cell_function=QCAD_CELL_NORMAL
number_of_dots=4
[TYPE:CELL_DOT]
x=64.500000
y=55.500000
diameter=5.000000
charge=8.010882e-020
```

```
spin=0.000000
potential=0.000000
[#TYPE:CELL_DOT]
[TYPE:CELL_DOT]
x=64.500000
y=64.500000
diameter=5.000000
charge=8.010882e-020
spin=0.000000
potential=0.000000
[#TYPE:CELL_DOT]
[TYPE:CELL_DOT]
x=55.500000
y=64.500000
diameter=5.000000
charge=8.010882e-020
spin=0.000000
potential=0.000000
[#TYPE:CELL_DOT]
[TYPE:CELL_DOT]
x=55.500000
y=55.500000
diameter=5.000000
charge=8.010882e-020
spin=0.000000
potential=0.000000
[#TYPE:CELL_DOT]
[#TYPE:QCADCell]
[#TYPE:QCADLayer]
[#TYPE:DESIGN]
```

Opis QCA celice vsebuje njen položaj, njene dimenzije, pripadajoč urin signal in podatke o vseh pripadajočih kvantnih pikah.

9.4 Uporaba aplikacije

Aplikacijo se zažene z izvedljivo datoteko ”place and route.exe”. Za vhodno datoteko se obravnava datoteka ”logic design.xml”, ki se nahaja v isti mapi kot izvedljiva datoteka.

Izvorna koda, izvedljiva datoteka in vsi v disertaciji navedeni zgledi so priloženi pričujočemu delu v elektronski obliki.

10 Zaključek

V doktorski disertaciji smo razvili avtomatizirani pristop k snovanju fizične razmestitve strukture kvantnih celičnih avtomatov. Pri tem smo postavili zahtevo, da naj bo zasnovano strukturo dejansko možno tehnološko realizirati. Zato smo preučili termodinamične dejavnike, ki vplivajo na delovanje QCA in ta dejstva upoštevali pri snovanju. Poleg tega smo obravnavali tudi implementacijske lastnosti QCA in pripadajočega vezja za generiranje urinega signala. V razvitem snovanju smo upoštevali njune značilnosti in zahteve, ki jih slednje postavljajo pri povezovanju QCA strukture in urinega vezja. Obravnava naštetih dejavnikov je nujna za zasnovo tehnološko izvedljivih struktur QCA.

Osredotočili smo se na fizično snovanje, razdeljeno na tri glavne stopnje: snovanje geometrije strukture, razmeščanje logičnih primitivov in povezovanje logičnih primitivov. V stopnji snovanja geometrije strukture smo z upoštevanjem lastnosti QCA in dejavnikov, ki vplivajo na njihovo delovanje, določili obliko urinih con in njihovo medsebojno razporeditev. Na podlagi topološko urejenega usmerjenega grafa logične sheme smo opredelili možne razdelitve logičnih primitivov v urine cone. Razvili smo geometrijo strukture, z uporabo katere je možno zasnovati izvedljive strukture QCA. Vendar pa obstoječi logični

primitivi QCA niso bili primerni za uporabo v tej geometriji. Ker so majoritetna vrata pogosto uporabljan logični primitiv QCA, smo jih nadgradili za uporabo v naši geometriji strukture. Njihovo pravilno delovanje smo potrdili s simulacijo v orodju QCADesigner in s tem pokazali, da je možno na podlagi razvite geometrije z uporabo nadgrajenih logičnih primitivov zasnovati poljubno kombinatorično vezje v kvantnih celičnih avtomatih.

V stopnji razmeščanja se določijo položaji logičnih primitivov znotraj pripadajoče urine cone. Postavili smo načrtovalska pravila, ki opredeljujejo potrebne razdalje med logičnimi primitivi za zagotovitev pravilnega delovanja celotne strukture QCA. Za razmeščanje smo uporabili metodo simuliranega ohlajanja, pri čemer smo optimizirali vsoto dolžin vseh linij in število križanj linij. Optimizirana zasnova strukture QCA je manj kompleksna, zato omogoča lažjo tehnološko izvedbo.

V zadnji stopnji fizičnega snovanja se ustrezni logični primitivi med seboj povežejo z linijami. Tudi tu smo postavili načrtovalska pravila, ki definirajo potrebne razdalje med posameznimi linijami ter med linijami in logičnimi primitivi. Poleg tega pravila določajo ustrezne razdalje med križajočimi se linijami. Če te razdalje niso ustrezne, je potrebno vstaviti dodatno urino cono in vanjo prenesti določen del linije. Le tako je delovanje strukture QCA pravilno, kar dokazujejo rezultati simulacij v orodju QCADesigner. Logične primitive povežemo z uporabo kombinacije metode iskanja po labirintu in metode iskanja po liniji. Tako optimiziramo vsoto dolžin vseh linij in število kotnih linij, kar vodi do manjše kompleksnosti zasnovane strukture QCA.

V pričujočem delu smo formalno opisali postopek fizičnega snovanja razmestitve strukture QCA. Razviti postopek upošteva fizikalne značilnosti QCA, zato lahko z njegovo uporabo zasnujemo tehnološko izvedljive strukture QCA. Med snovanjem se upoštevajo natančno definirana načrtovalska pravila, kar zagotavlja pravilno delovanje strukture. Formalizacija postopka omogoča njegovo avtomatizacijo, kar smo potrdili z implementacijo orodja za računalniško podprto snovanje fizične razmestitve strukture QCA. Avtomatiziran postopek omogoča lažje in hitrejše snovanje velikih struktur QCA, pri čemer je možnost pojave napake veliko manjša kot pri ročnem snovanju. Pravilno delovanje vseh zasnovanih struktur QCA, ki upoštevajo razvita načrtovalska pravila, je dokazano s simulacijo z uporabo koherenčnega vektorja v orodju QCADesigner.

Z uporabo nadgrajenih majoritetnih vrat, negatorjev in linij, je na podlagi razvite geometrije strukture možno zasnovati poljubno kombinatorično vezje. Nadaljnje delo bo vključevalo snovanje geometrije vezja, ki bo primerno za namestitev sekvenčnega vezja.

Pri tem bo potrebno določiti urine cone, v katerih bodo nameščene povratne povezave. Nadaljnje delo bo obsegalo tudi razširitev načrtovalskih pravil, ki bodo obravnavale urine cone, širše od šestih področij. Za avtomatizacijo postopka snovanja je možno implementirati dodatne metode in njihove rezultate primerjati z rezultati že razvitih metod.

LITERATURA

[1] R. R. Schaller, "Moore's law: past, present, and future," *IEEE Spectrum*, št. 34, zv. 6, str. 52–59, jun. 1997.

[2] International Technology Roadmap for Semiconductors, "2009 Edition," tehnično poročilo, Semiconductor Industry Association, 2009.

[3] C. S. Lent, P. D. Tougaw, W. Porod, G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, št. 4, zv. 1, str. 49–57, jan. 1993.

[4] P. D. Tougaw, C. S. Lent, "Logical devices implemented using quantum cellular automata," *Journal of Applied Physics*, št. 75, zv. 3, str. 1818–1825, feb. 1994.

[5] H. Cho, E. E. Swartzlander, "Adder Designs and Analyses for Quantum-Dot Cellular Automata," *IEEE Transactions on Nanotechnology*, št. 6, zv. 3, str. 374–383, maj 2007.

[6] H. Cho, E. E. Swartzlander, "Adder and Multiplier Design in Quantum-Dot Cellular Automata," *IEEE Transactions on Computers*, št. 58, zv. 6, str. 721–727, jun. 2009.

[7] C. Mead, L. Conway, *Introduction to VLSI Systems*. Addison-Wesley, 1979.

[8] N. Zimic, J. Virant, *Logično načrtovanje računalniških struktur in sistemov*. Fakulteta za računalništvo in informatiko v Ljubljani, 1998.

[9] W. Wolf, *Modern VLSI Design*. Pearson Education, Inc., 2009.

[10] L.-T. Wang, Y.-W. Chang, K.-T. T. Cheng, eds., *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann Publishers, 2009.

[11] C. J. Alpert, D. P. Mehta, S. S. Sapatnekar, eds., *Handbook of Algorithms for Physical Design Automation*. Auerbach Publications, 2009.

[12] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, št. 220, zv. 4598, str. 671–680, maj 1983.

[13] A. V. Cabot, R. L. Francis, M. A. Stary, "A Network Flow Solution to a Rectilinear Distance Facility Location Problem," *AIIE Transactions*, št. 2, zv. 2, str. 132–141, jun. 1970.

[14] J. M. Kleinhans, G. Sigl, F. M. Johannes, K. Antreich, "GORDIAN: VLSI placement by quadratic programming and slicing optimization," *IEEE Transactions on Computer-Aided Design*, št. 10, zv. 3, str. 356–365, mar. 1991.

[15] C. J. Fisk, D. L. Caskey, L. E. West, "ACCEL: Automated Circuit Card Etching Layout," *Proceedings of the IEEE*, št. 55, zv. 11, str. 1971–1982, nov. 1967.

[16] M. A. Breuer, "A class of min-cut placement algorithms," v zborniku *Proceedings of the 14th Design Automation Conference*, DAC '77, (Piscataway, NJ, USA), str. 284–290, IEEE Press, 1977.

[17] C. Y. Lee, "An Algorithm for Path Connections and Its Applications," *IRE Transactions on Electronic Computer*, št. EC-10, zv. 3, str. 346–365, sept. 1961.

[18] P. E. Hart, N. J. Nilsson, B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, št. 4, zv. 2, str. 100–107, jul. 1968.

[19] M. T. Niemier, M. J. Kontz, P. M. Kogge, "A Design of and Design Tools for a Novel Quantum Dot Based Microprocessor," v zborniku *Proceedings of the 37th Annual Design Automation Conference*, str. 227–232, 2000.

[20] M. T. Niemier, "The effects of a new technology on the design, organization, and architectures of computing systems," doktorska disertacija, Graduate Program in Computer Science and Engineering Notre Dame, Indiana, 2003.

[21] R. Ravichandran, S. K. Lim, M. Niemier, "Automatic cell placement for quantum-dot cellular automata," *Integration, the VLSI Journal*, št. 38, zv. 3, str. 541–548, jan. 2005.

[22] M. Choi, Z. Patitz, B. Jin, F. Tao, N. Park, M. Choi, "Designing layout-timing independent quantum-dot cellular automata (QCA) circuits by global asynchrony," *Journal of Systems Architecture*, št. 53, zv. 9, str. 551–567, sept. 2007.

[23] T. Teodosio, L. Sousa, "QCA-LG: A tool for the automatic layout generation of QCA combinational circuits," v zborniku *25th IEEE Norchip Conference*, str. 1–5, 2007.

[24] M. Bubna, S. Roy, N. Shenoy, S. Mazumdar, "A layout-aware physical design method for constructing feasible QCA circuits," v zborniku *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, GLSVLSI '08, (New York, NY, USA), str. 243–248, ACM, 2008.

[25] M. Crocker, M. Niemier, X. S. Hu, M. Lieberman, "Molecular QCA design with chemically reasonable constraints," *ACM Journal on Emerging Technologies in Computing Systems*, št. 4, zv. 9, str. 9:1–9:21, apr. 2008.

[26] Y. Zheng, "Circuit Design Methods with Emerging Nanotechnologies," doktorska disertacija, Faculty of Virginia Polytechnic Institute and State University, 2009.

[27] C. S. Lent, P. Tougaw, "A Device Architecture for Computing with Quantum Dots," *Proceedings of the IEEE*, št. 85, zv. 4, str. 541–557, apr. 1997.

[28] M. T. Niemier, P. Kogge, "Problems in Designing with QCAs: Layout = Timing," *International Journal of Circuit Theory and Applications*, št. 29, zv. 1, str. 49–62, jan. 2001.

[29] K. Hennessy, C. S. Lent, "Clocking of molecular quantum-dot cellular automata," *Journal of Vacuum Science Technology B: Microelectronics and Nanometer Structures*, št. 19, zv. 5, str. 1752–1755, sept. 2001.

[30] W. J. Chung, B. Smith, S. K. Lim, "QCA Physical Design With Crossing Minimization," v zborniku *Proceedings of 2005 5th IEEE Conference on Nanotechnology*, str. 262–265, 2005.

[31] B. S. Smith, S. K. Lim, "QCA Channel Routing With Wire Crossing Minimization," v zborniku *GLSVLSI'05*, str. 217–220, 2005.

[32] A. Chaudhary, D. Z. Chen, X. S. Hu, M. T. Niemier, R. Ravichandran, K. Whitton, "Fabricatable Interconnect and Molecular QCA Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, št. 26, zv. 11, str. 1978–1991, nov. 2007.

[33] R. Devadoss, K. Paul, M. Balakrishnan, "Clocking-Based Coplanar Wire Crossing Scheme for QCA," v zborniku *Proceedings of the 2010 23rd International Conference on VLSI Design*, VLSID '10, (Washington, DC, USA), str. 339–344, IEEE Computer Society, 2010.

[34] S. C. Henderson, E. W. Johnson, J. R. Janulis, P. D. Tougaw, "Incorporating Standard CMOS Design Process Methodologies into the QCA Logic Design Process," *IEEE Transactions on Nanotechnology*, št. 3, zv. 1, str. 2–9, mar. 2004.

[35] P. Tougaw, C. S. Lent, "Dynamic behavior of quantum cellular automata," *Journal of Applied Physics*, št. 80, zv. 8, str. 4722–4736, okt. 1996.

[36] G. Schulhof, K. Walus, G. A. Jullien, "Simulation of random cell displacements in QCA," *ACM Journal on Emerging Technologies in Computing Systems*, št. 3, zv. 1, str. 2:1–2:14, apr. 2007.

[37] K. Walus, T. J. Dysart, G. A. Jullien, R. A. Budiman, "QCADesigner: a rapid design and Simulation tool for quantum-dot cellular automata," *IEEE Transactions on Nanotechnology*, št. 3, zv. 1, str. 26–31, mar. 2004.

[38] P. Venkataramani, S. Srivastava, S. Bhanja, "Sequential Circuit Design in Quantum-Dot Cellular Automata," v zborniku *8th IEEE Conference on Nanotechnology, 2008. NANO '08*, str. 534–537, 2008.

[39] M. Mahdavi, M. A. Amiri, S. Mirzakuchaki, M. N. Moghaddasi, "Single Electron Fault in QCA Inverter Gate," v zborniku *Fifth International Conference on MEMS, NANO, and Smart Systems (ICMENS)*, str. 63–66, 2009.

[40] M. Macucci, ed., *Quantum cellular automata: theory, experimentation and prospects*. Imperial College Press, 2006.

[41] T. Toffoli, N. Margolus, *Cellular automata machines: a new environment for modeling*. Cambridge, MA, USA: MIT Press, 1987.

[42] J. R. Weimar, *Simulation with Cellular Automata*. Logos Verlag Berlin, 1997.

[43] J. Timler, C. S. Lent, "Power gain and dissipation in quantum-dot cellular automata," *Journal of Applied Physics*, št. 91, zv. 2, str. 823–831, jan. 2002.

[44] R. K. Kummamuru, J. Timler, G. Toth, C. S. Lent, R. Ramasubramaniam, A. O. Orlov, G. H. Bernstein, G. L. Snider, "Power gain in a quantum-dot cellular automata latch," *Applied Physics Letters*, št. 81, zv. 7, str. 1332–1334, aug. 2002.

[45] M. Alam, M. Siddiq, G. Bernstein, M. Niemier, W. Porod, X. Hu, "On-Chip Clocking for Nanomagnet Logic Devices," *IEEE Transactions on Nanotechnology*, št. 9, zv. 3, str. 348–351, maj 2010.

[46] S. Frost, T. Dysart, P. Kogge, C. Lent, "Carbon nanotubes for quantum-dot cellular automata clocking," v zborniku *4th IEEE Conference on Nanotechnology*, str. 171–173, 2004.

[47] G. Toth, C. S. Lent, "Quasiadiabatic switching for metal-island quantum-dot cellular automata," *Journal of Applied Physics*, št. 85, zv. 5, str. 2977–2984, mar. 1999.

[48] A. O. Orlov, I. Amlani, R. K. Kummamuru, R. Ramasubramaniam, G. Toth, C. S. Lent, G. H. Bernstein, G. L. Snider, "Experimental demonstration of clocked single-electron switching in quantum-dot cellular automata," *Applied Physics Letters*, št. 77, zv. 2, str. 295–297, jul. 2000.

[49] C. S. Lent, B. Isaksen, "Clocked molecular quantum-dot cellular automata," *IEEE Transactions on Electron Devices*, št. 50, zv. 9, str. 1890–1896, aug. 2003.

[50] G. Bernstein, A. Imre, V. Metlushko, A. Orlov, L. Zhou, L. Ji, G. Csaba, W. Porod, "Magnetic QCA systems," *Microelectronics Journal*, št. 36, zv. 7, str. 619–624, jul. 2005. European Micro and Nano Systems - EMN 2004.

[51] A. Imre, G. Csaba, L. Ji, A. Orlov, G. H. Bernstein, W. Porod, "Majority Logic Gate for Magnetic Quantum-Dot Cellular Automata," *Science*, št. 311, zv. 5758, str. 205–208, jan. 2006.

[52] K. Walus, G. Jullien, V. Dimitrov, "Computer arithmetic structures for quantum cellular automata," v zborniku *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, 2003.*, str. 1435–1439, 2003.

[53] G. Bazan, A. O. Orlov, G. L. Snider, G. H. Bernstein, "Charge detector realization for AlGaAs/GaAs quantum-dot cellular automata," *Journal of Vacuum Science Technology B: Microelectronics and Nanometer Structures*, št. 14, zv. 6, str. 4046–4050, nov. 1996.

[54] S. Gardelis, C. G. Smith, J. Cooper, D. A. Ritchie, E. H. Linfield, Y. Jin, "Evidence for transfer of polarization in a quantum dot cellular automata cell consisting of semiconductor quantum dots," *Physical Review B*, št. 67, zv. 3, str. 033302–1–033302–4, jan. 2003.

[55] M. Mitic, M. C. Cassidy, K. D. Petersson, R. P. Starrett, E. Gauja, R. Brenner, R. G. Clark, A. S. Dzurak, C. Yang, D. N. Jamieson, "Demonstration of a silicon-based quantum cellular automata cell," *Applied Physics Letters*, št. 89, zv. 1, str. 013503–1–013503–3, jul. 2006.

[56] C. S. Lent, P. D. Tougaw, "Bistable saturation due to single electron charging in rings of tunnel junctions," *Journal of Applied Physics*, št. 75, zv. 8, str. 4077–4080, apr. 1994.

[57] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, G. L. Snider, "Realization of a Functional Cell for Quantum-Dot Cellular Automata," *Science*, št. 277, zv. 5328, str. 928–930, aug. 1997.

[58] I. Amlani, A. O. Orlov, G. L. Snider, C. S. Lent, G. H. Bernstein, "External charge state detection of a double-dot system," *Applied Physics Letters*, št. 71, zv. 12, str. 1730–1732, sept. 1997.

[59] G. L. Snider, A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, J. L. Merz, W. Porod, "Experimental demonstration of quantum-dot cellular automata," *Semiconductor Science and Technology*, št. 13, zv. 8A, str. A130–A134, aug. 1998.

[60] I. Amlani, A. O. Orlov, G. Toth, G. H. Bernstein, C. S. Lent, G. L. Snider, "Digital Logic Gate Using Quantum-Dot Cellular Automata," *Science*, št. 284, zv. 5412, str. 289–291, apr. 1999.

[61] A. O. Orlov, I. Amlani, G. Toth, C. S. Lent, G. H. Bernstein, G. L. Snider, "Experimental demonstration of a binary wire for quantum-dot cellular automata," *Applied Physics Letters*, št. 74, zv. 19, str. 2875–2877, maj 1999.

[62] G. H. Bernstein, I. Amlani, A. O. Orlov, C. S. Lent, G. L. Snider, "Observation of switching in a quantum-dot cellular automata cell," *Nanotechnology*, št. 10, zv. 2, str. 166–173, jun. 1999.

[63] G. L. Snider, A. O. Orlov, I. Amlani, X. Zuo, G. H. Bernstein, C. S. Lent, J. L. Merz, W. Porod, "Quantum-dot cellular automata," *Journal of Vacuum Science Technology A: Vacuum, Surfaces, and Films*, št. 17, zv. 4, str. 1394–1398, jul. 1999.

[64] W. Porod, C. S. Lent, G. H. Bernstein, A. O. Orlov, I. Amlani, G. L. Snider, J. L. Merz, "Quantum-dot cellular automata: computing with coupled quantum dots," *International Journal of Electronics*, št. 86, zv. 5, str. 549–590, maj 1999.

[65] C. S. Lent, "Bypassing the Transistor Paradigm," *Science*, št. 288, zv. 5471, str. 1597–1599, jun. 2000.

[66] C. S. Lent, B. Isaksen, M. Lieberman, "Molecular Quantum-Dot Cellular Automata," *Journal of the American Chemical Society*, št. 125, zv. 4, str. 1056–1063, jan. 2003.

[67] R. P. Cowburn, M. E. Welland, "Room Temperature Magnetic Quantum Cellular Automata," *Science*, št. 287, zv. 5457, str. 1466–1468, feb. 2000.

[68] A. Orlov, A. Imre, G. Csaba, L. Ji, W. Porod, G. H. Bernstein1, "Magnetic Quantum-Dot Cellular Automata: Recent Developments and Prospects," *Journal of Nanoelectronics and Optoelectronics*, št. 3, zv. 1, str. 55–68, mar. 2008.

[69] A. Gin, P. D. Tougaw, S. Williams, "An alternative geometry for quantum-dot cellular automata," *Journal of Applied Physics*, št. 85, zv. 12, str. 8281–8286, jun. 1999.

[70] T. Cole, J. C. Lusth, "Quantum-dot cellular automata," *Progress in Quantum Electronics*, št. 25, zv. 4, str. 165–189, jan. 2001.

[71] M. Macucci, G. Iannaccone, S. Francaviglia, B. Pellegrini, "Semiclassical simulation of quantum cellular automaton cells," *International Journal of Circuit Theory and Applications*, št. 29, zv. 1, str. 37–47, jan. 2001.

[72] G. Toth, "Correlation and coherence in quantum-dot cellular automata," doktorska disertacija, Department of Electrical Engineering Notre Dame, Indiana, 2000.

[73] H. H. Loomis, R. H. Wyman, "On Complete Sets of Logic Primitives," *IEEE Transactions on Electronic Computers*, št. EC-14, zv. 2, str. 173–174, apr. 1965.

[74] G. Klir, "On Universal Logic Primitives," *IEEE Transactions on Computers*, št. 20, zv. 4, str. 467–469, apr. 1971.

[75] K. Walus, G. Jullien, "Design Tools for an Emerging SoC Technology: Quantum-Dot Cellular Automata," *Proceedings of the IEEE*, št. 94, zv. 6, str. 1225–1244, jun. 2006.

[76] S. Frost, A. Rodrigues, A. Janiszewski, R. Rausch, P. Kogge, "Memory in Motion: A Study of Storage Structures in QCA," v zborniku *First Workshop on Non-Silicon Computation (NSC-1)*, str. 30–37, 2002.

[77] C. S. Lent, P. D. Tougaw, "Lines of interacting quantum-dot cells: A binary wire," *Journal of Applied Physics*, št. 74, zv. 10, str. 6227–6233, nov. 1993.

[78] L. Bonci, M. Gattobigio, G. Iannaccone, M. Macucci, "Monte-Carlo Simulation of Clocked and Non-Clocked QCA Architectures," *Journal of Computational Electronics*, št. 1, zv. 1, str. 49–53, jul. 2002.

[79] S. Bhanja, S. Sarkar, "Probabilistic Modeling of QCA Circuits Using Bayesian Networks," *IEEE Transactions on Nanotechnology*, št. 5, zv. 6, str. 657–670, nov. 2006.

[80] S. Srivastava, S. Bhanja, "Hierarchical Probabilistic Macromodeling for QCA Circuits," *IEEE Transactions on Computers*, št. 56, zv. 2, str. 174–190, feb. 2007.

[81] M. T. Niemier, P. M. Kogge, "Exploring and exploiting wire-level pipelining in emerging technologies," v zborniku *Special Issue: Proceedings of the 28th annual international symposium on Computer architecture (ISCA '01)*, (New York, NY, USA), str. 166–177, ACM, 2001.

[82] M. T. Niemier, A. F. Rodrigues, P. M. Kogge, "A Potentially Implementable FPGA for Quantum Dot Cellular Automata," v zborniku *1st Workshop on Non-Silicon Computation (NSC-1)*, str. 38–45, 2002.

[83] S. Haruehanroengra, W. Wang, "Efficient Design of QCA Adder Structures," *Solid State Phenomena*, št. 121-123, zv. Nanoscience and Technology, str. 553–556, mar. 2007.

[84] D. A. Antonelli, D. Z. Chen, T. J. Dysart, X. S. Hu, A. B. Kahng, P. M. Kogge, R. C. Murphy, M. T. Niemier, "Quantum-Dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions," v zborniku *Proceedings of the 41st annual Design Automation Conference*, str. 363–368, 2004.

[85] S. K. Lim, R. Ravichandran, M. Niemier, "Partitioning and Placement for Buildable QCA Circuits," *ACM Journal on Emerging Technologies in Computing Systems*, št. 1, zv. 1, str. 50–72, apr. 2005.

[86] A. Chaudhary, D. Z. Chen, K. Whitton, M. Niemier, R. Ravichandran, "Eliminating wire crossings for molecular quantum-dot cellular automata implementation," v zborniku *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, ICCAD '05, (Washington, DC, USA), str. 565–571, IEEE Computer Society, 2005.

[87] V. Vankamamidi, M. Ottavi, F. Lombardi, "Clocking and Cell Placement for QCA," v zborniku *Sixth IEEE Conference on Nanotechnology, 2006. IEEE-NANO 2006.*, vol. 1, str. 343–346, 2006.

[88] T. T. Teodosio, "Computacao Quantica: arquitecturas e simulacao de operacao de dispositivos," magistrska naloga, Engenharia Electrotecnica e de Computadores, 2007.

[89] V. Vankamamidi, M. Ottavi, F. Lombardi, "Two-Dimensional Schemes for Clocking/Timing of QCA Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, št. 27, zv. 1, str. 34–44, jan. 2008.

[90] A. Gin, S. Williams, H. Meng, P. D. Tougaw, "Hierarchical design of quantum-dot cellular automata devices," *Journal of Applied Physics*, št. 85, zv. 7, str. 3713–3720, apr. 1999.

[91] J. Huang, M. Momenzadeh, F. Lombardi, "Design of sequential circuits by quantum-dot cellular automata," *Microelectronics Journal*, št. 38, zv. 4-5, str. 525–537, apr. 2007.

[92] V. Vankamamidi, M. Ottavi, F. Lombardi, "A Serial Memory by Quantum-Dot Cellular Automata (QCA)," *IEEE Transactions on Computers*, št. 57, zv. 5, str. 606–618, maj 2008.

[93] V. A. Mardiris, I. G. Karafyllidis, "Design and simulation of modular 2n to 1 quantum-dot cellular automata (QCA) multiplexers," *International Journal of Circuit Theory and Applications*, št. 38, zv. 8, str. 771–785, okt. 2010.

[94] International Technology Roadmap for Semiconductors, "Interconnect," tehnično poročilo, Semiconductor Industry Association, 2009.

[95] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1990.

[96] M. Janež, P. Pečar, M. Mraz, "Layout design of manufacturable quantum-dot cellular automata," *Microelectronics Journal*, 2012, DOI:10.1016/j.mejo.2012.03.007.

[97] J. C. Lusth, B. Dixon, "A characterization of important algorithms for quantum-dot cellular automata," *Information Sciences: an International Journal*, št. 113, zv. 3-4, str. 193–204, feb. 1999.

[98] T. J. Dysart, "It's all about the signal routing: Understanding the reliability of QCA circuits and systems," doktorska disertacija, Graduate Program in Computer Science and Engineering Notre Dame, Indiana, 2009.

[99] B. Korte, J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 2007.

[100] K. Mikami, K. Tabuchi, "A computer program for optimal routing of printed circuit connectors," v zborniku *Proc. Int. Federation for Information Processing H47*, str. 1475–1478, 1968.

[101] D. W. Hightower, "A solution to line-routing problems on the continuous plane," v zborniku *DAC '69: Proceedings of the 6th annual Design Automation Conference*, (New York, NY, USA), str. 1–24, ACM, 1969.

[102] M. Mahdavi, M. Amiri, S. Mirzakuchaki, "Single Electron Fault in QCA Binary Wire," v zborniku *Second International Conference on Advances in Circuits, Electronics and Micro-electronics, 2009. CENICS '09*, str. 8–10, 2009.