Pregled delovanja tQCA struktur ter njihova implementacija v USE shemi

Andrej Dolenc[1,2,3], **Tilen Matkovič**[1,4], **and Rok Ivanšek**[1,2,5]

[1]Fakulteta za računalništvo in informatiko, Ljubljana, Slovenija
[2]Fakulteta za matematiko in fiziko, Ljubljana, Slovenija
[3]andrej.dolenc@student.uni-lj.si
[4]matkovic.tilen@gmail.com
[5]ivansek.rok@gmail.com

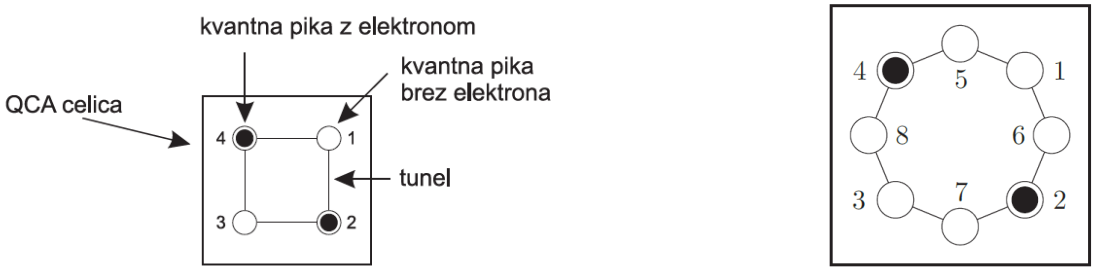
POVZETEK

Na preprost način lahko dvojiške kvantne celične avtomate nadgradimo v trojiške, ki jih uporabimo za implementacijo trojiških logik. Relativno malo pozornosti je bilo do sedaj namenjeno temu, kako se strukture izdelane iz takšnih trojiških celičnih avtomatov obnašajo pri različnih nastavitvah. V delu s pomočjo simulatorja testiramo nekaj osnovnih trojiških QCA struktur, kot so linije, križanje linij, pomnilniška celica, negator, majoritetna vrata in karakteristične funkcije. Za vsako od struktur pri 5 vnaprej določenih arhitekturah poiščemo, pri katerih razdaljah med celicami še delujejo pravilno. Nato predstavimo modifikacije teh struktur, da delujejo v fiksnem polju v katerem si večji skupki celic kvadratne oblike delijo isto urino periodo, in s pomočjo tako izdelanih gradnikov predstavimo še implementaciji karakteristične funkcije ter implikacije.

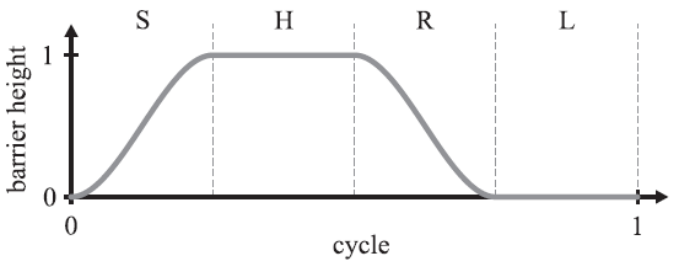
Uvod

Procesorji postajajo vedno manjši in se glede na njihovo trenutno zasnovo približujejo svojim fizičnim mejam. Zaradi tega bo v bližnji prihodnosti potrebno poseči po alternativnih tehnologijah, in ena izmed takšnih možnih tehnologij so t.i. kvantni celični avtomati (angl. Quantum-dot Cellular Automata, QCA), sestavljeni iz kvantnih celic.

Kvantna ali QCA celica je dvodimenzionalne površinske kvadratne oblike, na kateri se pri vsakem oglišču nahaja kvantna pika s pozitivnim nabojem[1]. V celici sta ujeta dva elektrona, ki se na podlagi fizikalnih odbojnih sil stabilizirata v eno izmed dveh možnih stanj. Na levi sliki 1 je razvidna shema dvojiške QCA celice. Z vpeljavo ure lahko začasno zaklepamo posamezne celice in s tem začasno onemogočimo prehajanje elektronov. Vsaka celica je s stališča ure lahko v enem izmed štirih možnih stanj, in sicer *Switch*, *Hold*, *Release* in *Relax*. S pomočjo ure 'potiskamo' stanje trenutnih celic na celice naslednje urine periode. Primer urinega signala je razviden iz slike 2.



Slika 1. Shema binarne (levo) in trojiške (desno) QCA celice.



Slika 2. Spreminjanje urinega signala skozi čas.

Preprosto nadgradnjo binarnih QCA celic lahko dosežemo tako, da v vsako celico dodamo še 4 dodatne kvantne pike (glej desno sliko 1), s čimer lahko vsako celico uporabimo za predstavitev do štirih različnih vrednosti[2]. Tako lahko, če dve izbrani stanji celice združimo v eno, izdelamo t.i. tQCA celice, ki jih lahko uporabimo za implementacijo trojiške logike po Łukasiewiczu. V odvisnosti od tega, kolikšne so razdalje med kvantnimi pikami znotraj posamezne celice ločimo načeloma 3 različne arhitekture: 60, 72, ter 110. Poleg teh uporabljamo še arhitekturi 91 ter 92, ki imata na sredi celice še dodatno piko, s čimer naj bi dosegli dodatno stabilnost. Pri združevanju celic v večje celote so pomembne še razdalje med celicami samimi.

To poročilo je sestavljeno iz dveh delov: v prvem delu s pomočjo simulatorja poskusimo odkriti meje delovanja osnovnih (že znanih) struktur v tQCA glede na uporabljeno arhitekturo. V drugem delu nato po vzoru članka Campos in sod.[3] izbrane strukture modificiramo do te mere, da jih lahko uporabimo v prostoru razdeljenem glede na vnaprej določene kvadrate z istimi urinimi fazami. Za demonstracijo delovanja koncepta za konec prikažemo še, kako v takšnem modelu naredimo strukturo za logično implikacijo v trojiški logiki po Łukasiewiczu.

Sorodno delo

Lebar Bajec, Zimic in Mraz[2], za katere bi lahko rekli da so začetniki trojiške QCA logike, so predstavili Łukasiewiczeve trovrednostne logične operacije negacije, disjunkcije in konjunkcije. Magdevski[4] je to delo nadaljeval in poskušal razviti Łukasiewiczevo trovrednostno implikacijo, kar mu je tudi uspelo. Dobil je tudi Łukasiewiczevi trovrednosti operaciji (disjunkcijo in konjunkcijo).

Poleg osnovnih trojiških QCA struktur (negacija, OR, AND) in majoritetne strukture, katere slabost je ta da zavzame relativno veliko prostora, je Mrazu in sod.[5] uspelo sestaviti po velikosti manjšo, t.i. *AO tQCA* strukturo, ki se lahko uporabi za implementacijo trojiške disjunkcije in konjunkcije.

Zanimiv dosežek je prišel še iz strani Janeža in sod.[6]. Ti so z uporabo metode iterativnega poglabljanja zasnovali postopek, ki avtomatično najde realizacijo poljubne logične funkcije z uporabo skoraj minimalnega števila vnaprej definiranih struktur.

Z adiabatno kontroliranimi osnovnimi strukturami linije, kotne linije, negatorja in majoritetnih vrat so Pečar in sod.[7] predstavili idejo, ki rešuje probleme nekaterih trojiških funkcij. Ideja je ta, da je poleg same strukture še urin signal, ki kontrolira katere celice so v fazah zadrževanja, preklopa, sproščenosti in sproščanja. Isti avtorji in sod.[8–10] so predstavili še adiabatni cevovod, ki rešuje nekatere probleme elementarne logike trojiškega QCA, kot so naprimer kotne linije in majoritetna vrata. Z omenjenim adiabatnim preklapljanjem in cevovodom, avtorji opozarjajo na možnosti implementacije naprednejših trojiških aritmetično logičnih in pomnilnih enot. Z osnovnimi pomnilnimi enotami je možno hraniti en trit. Te enote so ključne za hranjenje podatkov, s katerimi lahko gradimo kompleksnejše platforme procesiranja.

Pečar in Lebar Bajec[11] sta predstavila trojiško karakteristično funkcijo. Ta skupaj s trojiškimi QCA majoritetnimi vratmi in trojiškimi konstantami sestavlja funkcijsko poln nabor, katerega do zdaj s primitivno logiko trojiškega QCA (majoritetna vrata in negator), ni bilo možno sestaviti.

Študenti pri predmetu Nekonvencionalne metode in platforme procesiranja (in pri sorodnih predmetih) na Fakulteti za računalništvo in informatiko prejšnjih let so z več ali manj sreče uspešno implementirali nekaj trojiških QCA struktur. Ena od teh je implikacija v tQCA, ki pa je bila prezahtevna[12]. Podoben neuspeh so imeli tudi študentje[13], ki so iskali komparator oz. primerjalnik. Več sreče je imela skupina[14], ki je implementirala trojiško pomnilno celico, t.i. *WD celico*, za katero pravijo, da za razliko od sorodnejših *RS* in *JK* celic, lahko hrani več informacije. Z večnivojsko arhitekturo trojiških QCA struktur so se ukvarjali študenti[15], ki so preizkušali razne prenose čez več nivojev, križanje linij, premostitvijo in prišli do različnih zaključkov. Božič in sod.[16] so pokazali, da uporaba arhitekture, kjer so v celici kvantne pike bliže središču celice, vpliva na pravilnost končnega rezultata različnih struktur. Kolegom Lampreht in sod.[17] ni uspelo realizirati preprostega seštevalnika s trojiškimi QCA. Podobno Rolihu[18] ni uspelo najti strukture, ki bi v svetu tQCA enot realizirala (reverzibilna) Toffolijeva vrata.

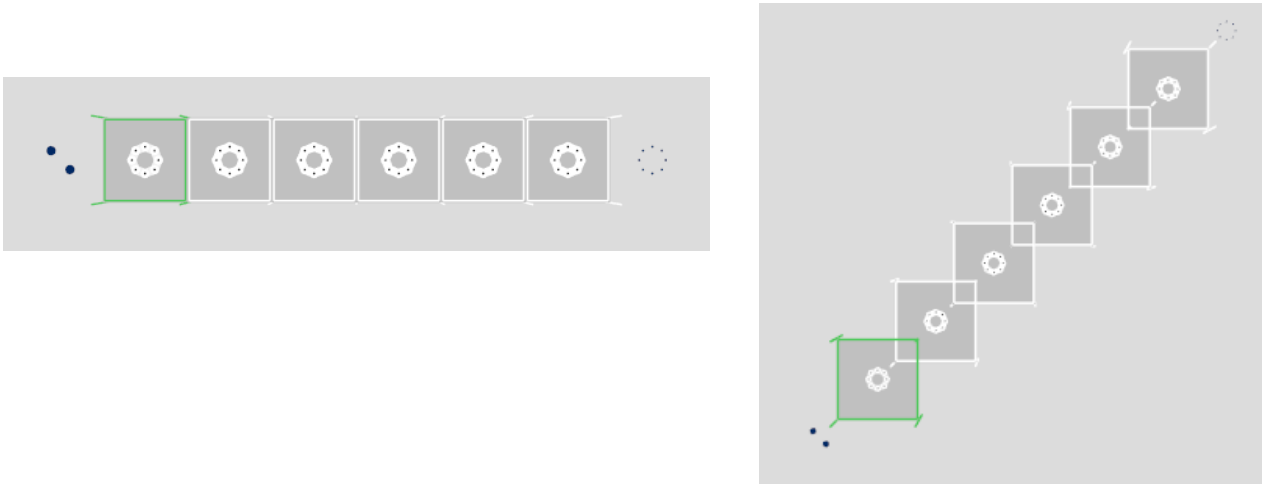
Pravilna polarizacija posameznih celic se kontrolira s pomočjo zunanjih urinih signalov, ki 'potiskajo' kvantne pike naprej. Pri večini objavljenih člankih ni bilo neke standarne urine sheme, ki bi uporabo posameznih celic oz. skupin celic naredila bolj

fleksibilno. Zaradi tega so Campos in sod.[3] predlagali t.i. USE (angl. *universal, scalable and efficient*) urino shemo, ki naj bi omogočala lažji razvoj QCA vezij. Avtorji članka so USE shemo uporabljali za dvojiški QCA v QCADesigner orodju, mi pa smo shemo uporabili za trojiški QCA, za katerega smo poskušali implementirati več struktur, opisanih v spodnjih poglavjih.

Osnovne strukture v tQCA

Za začetek smo preučili obnašanje struktur zgrajenih s tQCA celicami. Zanimalo nas je predvsem to, kako različne arhitekture ter razdalje med posameznimi celicami vplivajo na pravilnost končnega rezultata. Rezultate smo preverjali s pomočjo qdCad simulatorja verzije 1.8.20161018, ki so ga izdelali v laboratoriju LRSS Fakultete za računalništvo in informatiko Univerze v Ljubljani. Da smo našli intervale pravilnega delovanja v čim krajšem času smo uporabili metodo bisekcije. S takšnimi testi smo dobili občutek za nadalje konfiguracije, ki smo jih uporabili za snovanje kompleksnejših struktur.

Linije



Slika 3. Dva izbrana tipa linij, ki smo jih preučili.

Slika 3 prikazuje prvi dve takšni "prenašalni" strukturi, nad katerimi smo izvajali teste. Obe strukturi sta zasnovani tako, da za vsakega izmed vhodov $\{A,B,C\}$ na izhod postavita kar vhodno vrednost; gre torej za preprosto prenašanje signala. Poleg teh dveh linij smo preizkusili tudi vertikalno linijo, ter linijo, kjer se je signal prenašal preko z koordinate (torej med sloji).

Za testiranje struktur smo v simulatorju število urinih ciklov nastavili na 0.625, število evalvacij na posamezno fazo 50, minimalno spremembo E na $10^{-0.005}$, število korakov pa na 100. Tabele 1 prikazujejo intervale razdalj med celicami, ki vodijo do pravilnega delovanja v odvisnosti od arhitekture. Iz tabel je jasno razvidno, da premajhne razdalje med centri celic ponavadi vodijo v napačen končni rezultat, prav tako pa se s prevelikimi razdaljami začne večati možnost napak. Iz tabel pa ni razvidno, da simulator pri večjih razdaljah porabi občutno več časa za končanje simulacije (pri čemer se prav tako signifikantno poveča število 'epsDisregards'), medtem ko je bil čas izvajanja minimalen, ko je bila razdalja med celicami približno enako velika kot uporabljena arhitektura sama (oz. malenkost večja).

Tabela 1. Intervali razdalj med celicami ki vodijo do pravilnega delovanja horizontalne (oz. vertikalne) linije (levo), diagonalne linije (srednja tabela) ter linije po z koordinati (desno) v odvisnosti od arhitekture.

arhitektura	interval [nm]	
	od	do
60	66	398
72	82	451
110	120	573
91	75	367
92	117	284

arhitektura	interval [nm]	
	od	do
60	60	281
72	72	319
110	110	405
91	60	259
92	83	201

arhitektura	interval [nm]	
	od	do
60	60	212
72	72	240
110	110	305
91	60	240
92	60	305

Križanje linij

Križanje linij smo izvedli na primerih kot so razvidni na slikah 4-7. Pri križanju linij brez prenašanja informacije iz enega na drug nivo (slika 4), katerih ideja je bila vzeta od ene izmed lanskoletnih skupin[15], ni bilo večjih težav - pri dovolj veliki razdalji med posameznih nivojih ne pride do anomalij. Ena izmed delujočih arhitektur in razdalj med celicami obeh struktur je arhitektura 72 z razdaljami med celicami po x ter y osi 110 in 110, ter po z osi 55nm. Ista skupina je implementirala tudi križanje s premostitvijo (ena izmed implementacij je razvidna iz slike 5). Pri poskušanju različnih arhitektur in razdalj med celicami smo delujočo simulacijo dobili pri enakih razdaljah le da z uporabo arhitekture 91. Preizkusili smo tudi malenkost drugačno implementacijo, kjer je imel 'most' navpičen vzpon in spust, toda ni delovalo pravilno tudi v primeru ko smo nastavili vse parametre identične tistim iz originalnega članka.

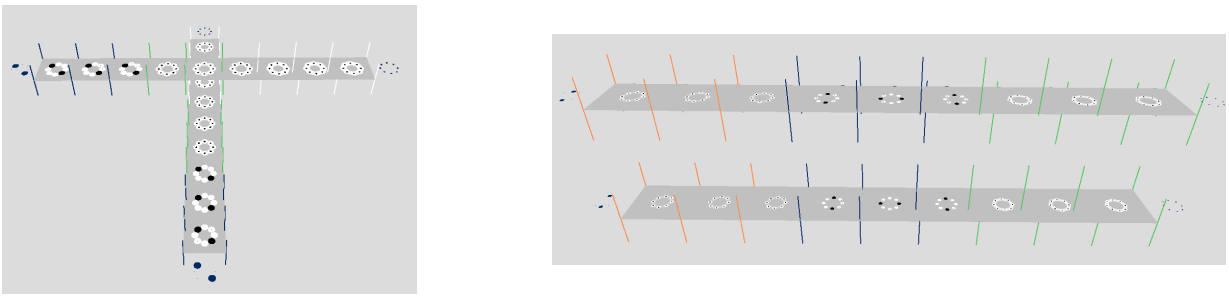
Tretji primer križanja linij je bil primarno izveden s strani Lebar Bajca & Pečarja[19] (glej sliko 6). Tu smo imeli probleme z nastavitvijo arhitekture, a z nekaj potrpežljivosti smo našli takšne, ki so dajale pravilne rezultate (vsaka simulacija je namreč trajala minimalno $\sim$ 5 minut). Rezultati so razvidni iz tabele 2. Po predlogu asistenta, da preizkusimo take z vrednosti, ki so za polovico manjše od ostalih razdalj, smo imeli manj sreče – napaka se je zgodila pri vhodu 'C B', kjer je simulator vrnil 'C C'. Nato smo z vrednosti malenkost zmanjšali (na takšne vrednosti, kot je opisano v opisu pri tabeli 2).

Tabela 2. Tabela levo prikazuje rezultate simulacij sinhroniziranega dvo-nivojskega prečkanja linij (glej sliko 6) v odvisnosti od arhitekture ter razdalje med celicami (pri zgornji tabeli sta x in y vrednosti tolikšni kot je navedeno, z pa za malenkost več kot polovico manjša - pri 60 je z razdalja 25, pri 72 30, pri 110 45, pri 160 pa 70). Tabela desno pa predstavlja rezultate simulacij pri x in y razdaljah '110', v odvisnosti od z-ja.

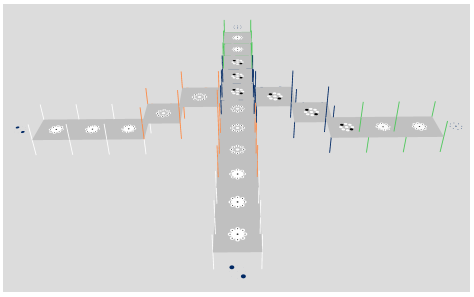
	arhitektura				
	60	72	110	91	92
60	x	x	x	x	x
72	x	x	x	x	x
110	1	1	x	1	x
160	1	1	x	1	x

z	arhitektura 72
35	x
40	x
42	1
45	1
55	x
80	x

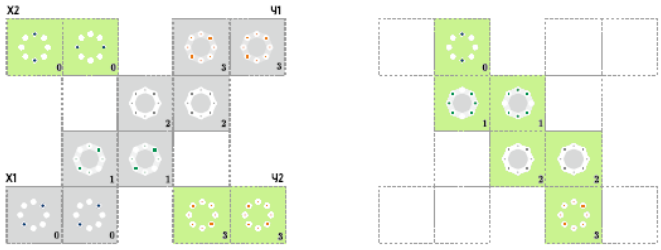
Zadnji primer je naš poskus križanja linij v ravnini (glej sliko 7). Ideja je ta, da najprej pošljemo informacijo po eni liniji, druga linija pa pošlje informacijo z 'zakasnjeno' urino periodo. Zaradi tega v točki križanja naj ne bi prišlo do konfliktov. Ena izmed nastavitev, pri kateri je delovanje pravilno, je na primer z arhitekturo 60 ter razdaljami med celicami 110 (brez z komponente, ki je tukaj nepomembna). Pri preizkušenih manjših vrednostih je v večini primerov prišlo do anomalij (nepravilen prenos na eni izmed linij). Ker takšnega prečkanja linij nismo uporabili v nadaljnem delu, natančnejšega ugotavljanja intervalov delovanja nismo opravljali.



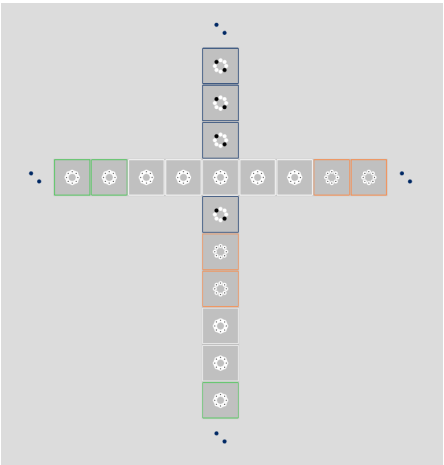
Slika 4. Dva tipa križanja linij, pri katerem je vsaka na svojem nivoju in ni prenašanja informacij prek nivojev.



Slika 5. Tip linij, kjer ena od linij prečka drugo (‘most’).



Slika 6. Sinhronizirano dvo-nivojsko prečkanje linij po Lebar Bajcu & Pečarju[19], leva shema predstavlja prvi nivo, desna drugi nivo, barvi pa določata različni liniji. Med navedenima nivojema je še en prazen nivo.

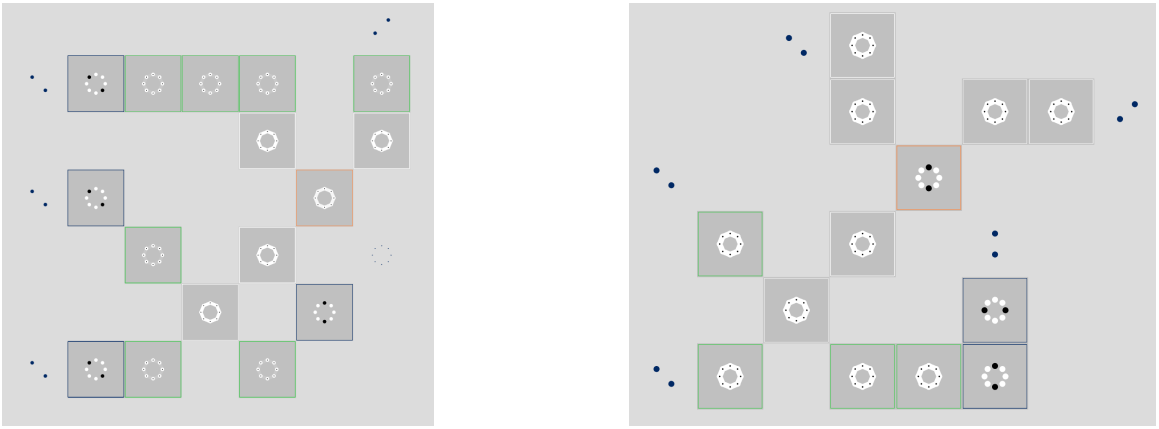


Slika 7. Naša implementacija križanja linij v ravnini (levo in spodaj sta vhoda, desno in zgoraj izhoda).

Pomnilniška celica

Implementirali smo pomnilniško celico, ki so jo predstavili Pečar in sod.[10]. Tukaj so prisotni štirje vhodi, in sicer prvi za podatek (A, B ali C), drugi za operacijo (za branje A in za pisanje B) in dve konstanti. Branje trenutne vrednosti v strukturi se izvede tako, da obe vrednosti (podatek in operacijo) postavimo na A. Delovanja strukture, kot je bila predstavljena v članku (levo na sliki 8), nam ni uspelo pravilno replicirati, tudi v primeru, ko so bili parametri enaki tistim iz članka. V eni od majoritetnih vrat je namreč prihajalo do anomalij. Zaradi tega smo strukturo prilagodili, kot je razvidno iz desne slike 8. Ta ima še zmeraj dvoje majoritetnih vrat in cevovod, torej ji nismo dodali ali odstranili nobene funkcionalnosti. Vhodi in izhodi strukture so razvidni iz tabele 4, delovanje pri določenih arhitekturah in razdaljah pa iz tabele 3. Ker smo dobili sumljive rezultate delovanja (struktura je namreč delovala na zelo velikem razponu), smo bili v dilemi, če je sploh pravilna. Takšne rezultate smo dobili verjetno zaradi tega, ker smo v neposredni bližini majoritetnih vrat postavili elektrodo, gonilno celico pa na enega izmed kotov vrat. Ko pa smo podaljšali linijo (torej ko smo elektrodo in gonilno celico postavili bolj stran, vmes pa

postavili notranje celice), smo vsakokrat dobili napačne rezultate. Elektroda in gonilna celica torej izhodno celico ‘silita’ na to, da ne spremeni svojega stanja in privedeta do takšnih rezultatov, kot smo jih dobili.



Slika 8. Implementacija pomnilniške celice iz originalnega članka[10](levo) in naša delujoča različica (desno). Pri obeh strukturah so vhodi/izhod sledeči (v smeri urinega kazalca, od spodaj levo): konstanta A, operacija (piši ali beri), podatek, konstanta B in izhod.

Tabela 3. Intervali razdalj med celicami ki vodijo do pravilnega delovanja naše delujoče različice pomnilniške celice. Opozorimo, da smo testirali le do razdalje 499, tako da pri arhitekturah 110 ter 92 lahko pravilno obnašanje dosežemo tudi pri večjih razdaljah.

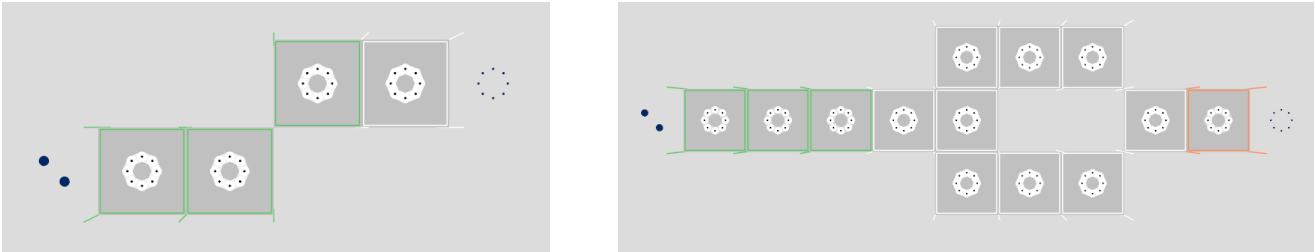
arhitektura	interval [nm]	
	od	do
60	60	408
72	72	462
110	110	499
91	60	467
92	90	499

Tabela 4. Vhodi in izhodi pri pomnilniški celici (glej sliko 8). Na začetku je v celici shranjen C.

x (vhodna vrednost)	w (beri ali piši)	konstanta B	konstanta A	izhod
A	A	B	A	C
A	B	B	A	A
A	A	B	A	A
B	B	B	A	B
A	A	B	A	B
C	B	B	A	C

Negator

Negator v trojiški logiki deluje tako, da vhod A zamenja za B, B za A, C pa pusti na isti vrednosti. Slika 9 prikazuje dve mogoči strukturi negatorja. Intervale delovanja leve strukture nismo posebej določali saj jih lahko pridobimo preko intervalov delovanja diagonalne ter vodoravne linije, alternativno strukturo pa smo preizkusili nad različnimi arhitekturami ter razdalji med celicami. Tabela 5 prikazuje rezultate simulacij.



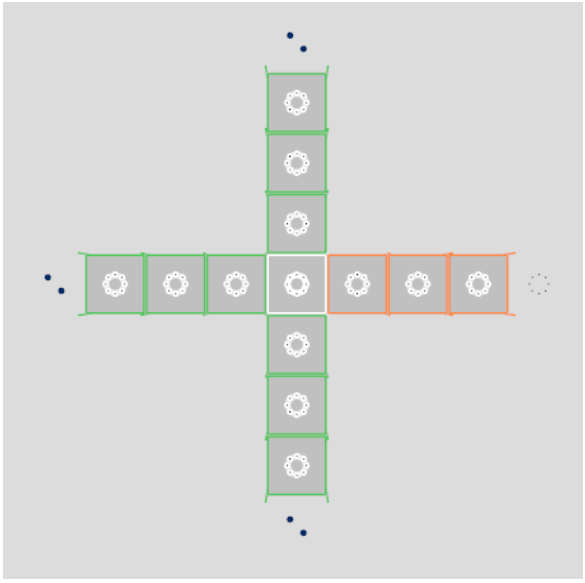
Slika 9. Osnovna (levo) ter alternativna struktura negatorja (desno).

Tabela 5. Intervali delovanja simulacij alternative strukture za negacijo v odvisnosti od arhitekture.

arhitektura	interval [nm]	
	od	do
60	104	273
72	144	313
110	153	408
91	121	355
92	159	415

Majoritetna vrata

Preizkušali smo tudi strukturo majoritetnih vrat, saj smo želeli izdelati strukturi za logični funkciji AND ter OR. Izkaže se, da lahko za implementacijo obeh funkcij uporabimo identično strukturo, le da na enega izmed vhodov v primeru AND postavimo A, v primeru OR pa B. Želeni vhodi in izhodi so predstavljeni v tabeli 7. Slika 10 prikazuje strukturo s katero smo uspešno simulirali obe funkciji, tabela 6 pa rezultate vseh pognanih simulacij.



Slika 10. Struktura majoritetnih vrat.

Tabela 6. Intervali pravilnega delovanja simulacij AND ter OR funkcij v odvisnosti od arhitekture.

arhitektura	interval [nm]	
	od	do
60	60	322
72	72	373
110	110	495
91	69	99
92	103	134

Tabela 7. Vhodi in želeni izhodi majoritetnih vrat.

vhod	izhod	vhod	izhod	vhod	izhod
A A A	A	A A B	A	A A C	A
A B A	A	A B B	B	A B C	C
A C A	A	A C B	C	A C C	C
B A A	A	B A B	B	B A C	C
B B A	B	B B B	B	B B C	B
B C A	C	B C B	B	B C C	C

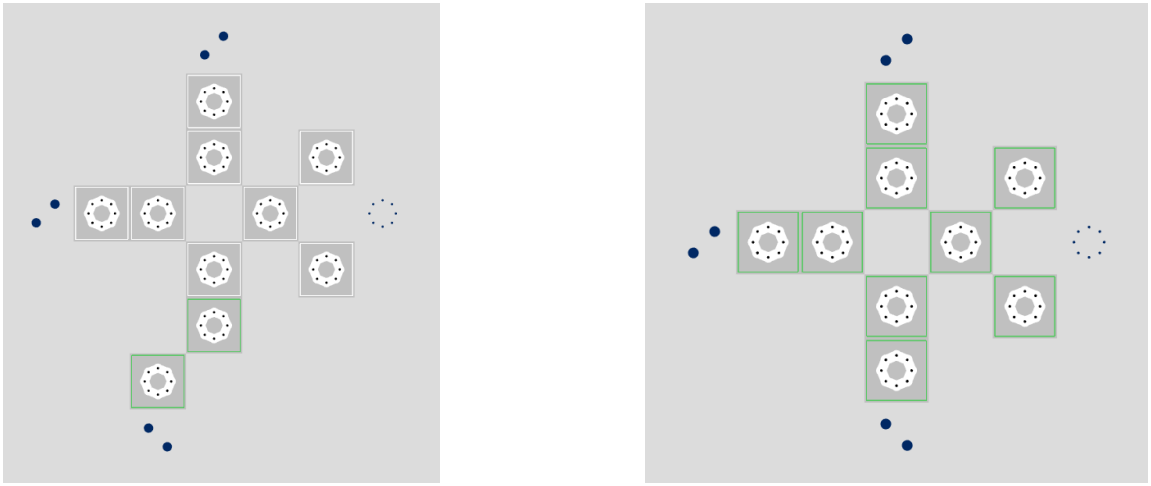
Karakteristične funkcije

Logične vrednosti, ki jih predstavljajo stanja trojiških kvantnih celic, lahko predstavimo z notacijo $\{-1,0,1\}$, kjer je mapiranje naslednje A $\rightarrow -1$, B $\rightarrow 1$ ter C/D $\rightarrow 0$. Glede na ta tri logična stanja lahko skonstruiramo tri karakteristične funkcije, t.j. funkcije, katerih obnašanje predstavimo z naslednjo enačbo:

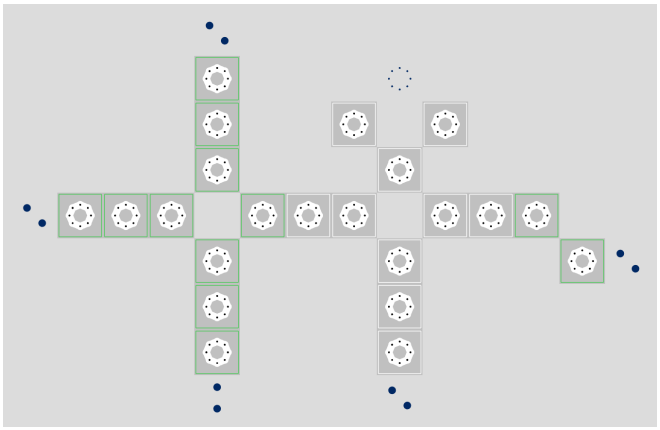
$$f^X(x) = \begin{cases} 1 & \text{,če } x = X, \\ -1 & \text{sicer.} \end{cases}$$

Kot je predstavljeno v članku[11] smo skonstruirali tri različne karakteristične funkcije, vsako za eno logično stanje. Vsaka od struktur na vhod prejme več različnih vrednosti, le ena pa predstavlja dejanski vhod strukture. Pričakovana obnašanja struktur za različne vhode so naslednja:

- f^1: ABB $\rightarrow$ A, BBB $\rightarrow$ B, CBB $\rightarrow$ A
- f^{-1}: ABB $\rightarrow$ B, BBB $\rightarrow$ A, CBB $\rightarrow$ A
- f^0: AAAAC $\rightarrow$ A, BAABC $\rightarrow$ A, DAADC $\rightarrow$ B



Slika 11. Karakteristični funkciji f^1 ter f^{-1}.



Slika 12. Karakteristična funkcija f^0.

Vidimo da je del vhoda fiksiran, del vhoda pa se spreminja. Spreminjajoči se del je tisti, ki predstavlja dejanski vhod za strukturo. Kot lahko vidimo na slikah 11 in 12, struktura f^{-1} obratuje v eni sami urini fazi, strukturi f^1 in f^0 pa za svoje delovanje potrebujeta dve urini fazi. V tabelah 8 lahko vidimo pri katerih arhitekturah in za katere razdalje se strukture obnašajo tako kot želimo.

Tabela 8. Rezultati simulacij karakteristične funkcije f^1 (levo), karakteristične funkcije f^{-1} (srednja tabela) in karakteristične funkcije f^0 (desno).

arhitektura	interval [nm]	
	od	do
60	60	280
72	72	318
110	110	406
91	60	314
92	78	401

arhitektura	interval [nm]	
	od	do
60	60	299
72	72	339
110	110	433
91	60	330
92	70	421

arhitektura	interval [nm]	
	od	do
60	60	245
72	72	278
110	110	355
91	84	186
92	131	279

Skupni intervali delovanja

Ker smo za vsako izmed struktur (razen za križanja linij) odkrili intervale razdalj med celicami, kjer le-te pravilno delujejo, lahko sedaj rezultate združimo in dobimo intervale, na katerih pri posamezni arhitekturi delujejo vse strukture; takšne intervale prikazuje tabela 9. Tu opozorimo na to, da se pri arhitekturah 91 ter 92 intervali delovanja majoritetnih vrat ne prekrivajo z intervali delovanja vseh ostalih struktur.

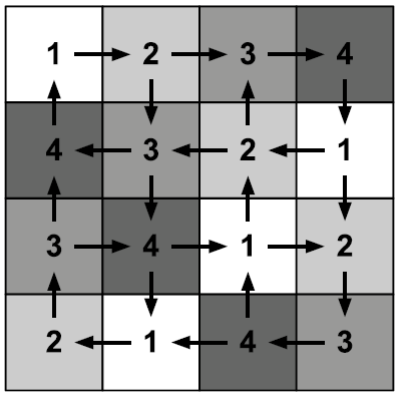
Tabela 9. Intervali razdalj med celicami ki vodijo do pravilnega delovanja vseh implementiranih struktur. Pri arhitekturah 91 ter 92 v predstavljenem intervalu delujejo vse strukture, razen majoritetnih vrat.

arhitektura	interval [nm]	
	od	do
60	104	212
72	144	240
110	153	305
91*	121	186
92*	158	201

USE shema

Kot je bilo predstavljeno v članku[3] za dvojiški QCA, smo tudi mi poskušali implementirati nekaj struktur, ki naj bi upoštevale USE urino shemo za trojiški QCA. Ideja je ta, da prostor razdelimo na 5x5 kvadrate, kjer vsak kvadrat predstavlja svojo urino periodo. Vsak od teh kvadratov mora biti na takšni lokaciji, da se signal prenaša le v sosednjo urino periodo (v smislu

npr. *HOLD* $\rightarrow$ *SWITCH*). Celotna ideja je lepše razvidna iz slike 13. S tem načinom zlaganja celic bi omogočili lažje in fleksibilnejše sestavljanje kompleksnejših struktur.



Slika 13. USE urina shema[3], ki prikazuje kvadrate urinih celic in prehode v vsakega od njih.

V preostanku poglavja predstavimo, katere strukture lahko nespremenjene uporabimo v USE urini shemi, ter modifikacije preostalim strukturam, da bodo delovale pravilno.

Nespremenjene strukture

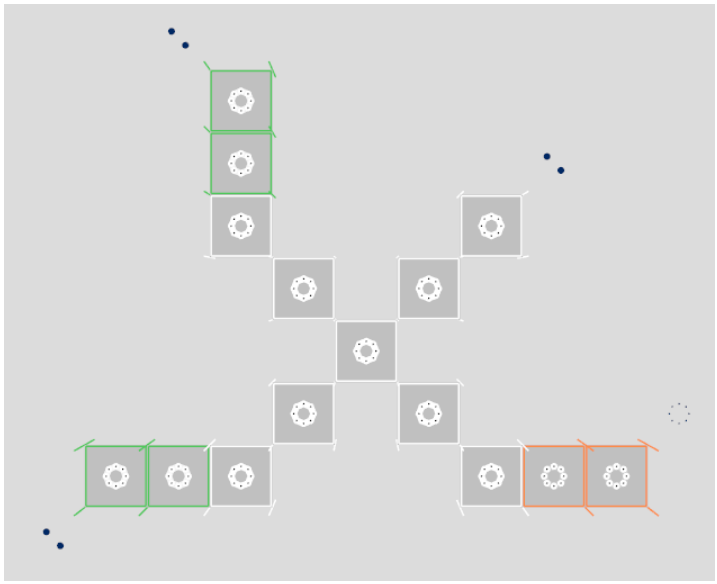
Očitno nespremenjene lahko ostanejo vse strukture linij, tako diagonalnih kot tudi horizontalnih. Prav tako lahko v 5x5 kvadratu v eni urini fazi nespremenjeno uporabimo alternativni model negatorja (ali pa osnovni model, če smo malce pazljivi), preostale strukture pa moramo nekoliko preurediti.

Na tej točki predlagamo, da namesto velikosti kvadrata 5x5 celic za tQCA raje uporabimo kvadrate velikosti 6x6. S tem lahko nespremenjeno v USE uporabimo tudi karakteristični funkciji f^1 ter f^{-1} (medtem ko moramo f^0 prilagoditi, saj trenutna struktura pričakuje vhodni signal iz dveh nasprotnih strani, kar v USE shemi ne moremo realizirati). Poleg tega se, če uporabimo sodo število celic izognemo tudi problemu, ki nastane pri linijah: vrednost 'C' se pri horizontalni liniji ob uporabi lihega števila celic namreč preslika v 'D' na drugi strani kvadrata (oz. v primeru diagonalne linije imamo problem s signaloma 'A' ter 'B').

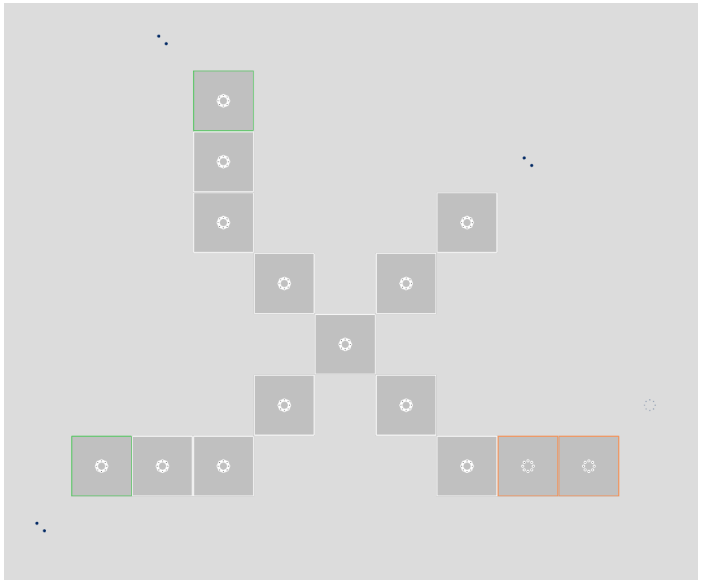
Majoritetna vrata

Majoritetna vrata, kot so ta predstavljena v prejšnjem poglavju, nam pri nobeni arhitekturi in razdalji med celicami ni uspelo pravilno prenesti v USE urino shemo. Problematično je namreč, ker morajo celotna vrata delovati v eni sami urini fazi, hkrati pa morajo vhodi in izhodi bili v drugih urinih fazah. Če izpolnimo te zahteve, lahko majoritetna vrata uporabimo kot samostojen 5x5 gradnik v večjih strukturah. Kot struktura, ki pod temi pogoji izkazuje pravilno delovanje majoritetnih vrat se je izkazala struktura, predstavljena na sliki 14. Takšna majoritetna vrata delujejo pravilno za arhitekturo 60 ter razdalje med celicami na intervalu $[170, 195]$ nm oz. za arhitekturo 72 pri razdaljah $[192, 222]$, pri ostalih arhitekturah pa ne delujejo pravilno.

Takšna majoritetna vrata lahko razširimo tudi na polje 6x6, kot je predstavljeno na sliki 15. Tudi ta vrata delujejo pravilno le za arhitekturi 60 (pri razdaljah $[165, 195]$) ter 72 (pri $[187, 217]$).



Slika 14. Struktura majoritetnih vrat za USE shemo 5x5.

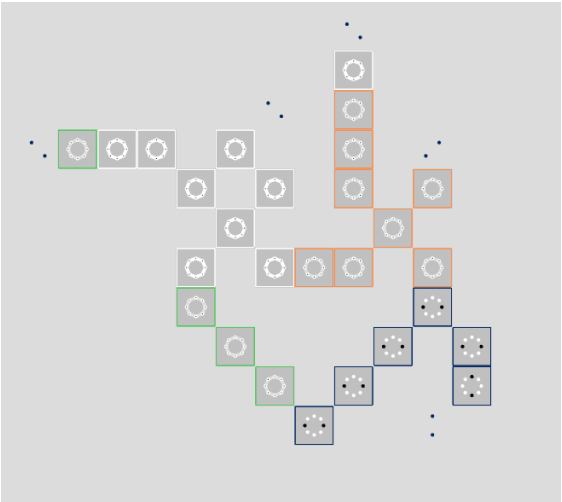


Slika 15. Struktura majoritetnih vrat za USE shemo 6x6.

Pomnilniška celica

Na sliki 16 je prikazana naša implementacija pomnilniške celice v 'raztegnjenem' 5x5 merilu po USE shemi, ki ima isto funkcionalnost kot navadna pomnilniška celica (glej sliko 8). Enako kot prejšnja različica ima tudi ta prav tako dvoje majoritetnih vrat, deluje pa vsaj pri arhitekturi 110 ter z razdaljami med celicami po 110 nm.

Vhodi in izhodi, ki smo jih morali dobiti, so bili enaki kot pri navadni pomnilniški celici (glej tabelo 4). Če bi že morali implementirati strukturo v 6x6 USE shemi, bi morali spremeniti le urini periodi pri dveh celicah, in sicer na sliki 16 najbolj levi in najbolj zgornji vhod (tako, da bi bili zelena in bela celica v urini periodi tistih sosednjih celic v katero smer potuje informacija). Analogija z navadno pomnilniško celico je ta naslednja: (na sliki 16) modra in zelena urina perioda predtavljata prenos informacije iz enih majoritetnih vrat na celoten izhod in na vhod na druga majoritetna vrata; bela urina urina perioda predstavlja ena majoritetna vrata, oranžna pa druga majoritetna vrata.



Slika 16. Pomnilniška celica v USE 5x5 shemi. Vhodi so sledeči (od leve proti desni): operacija beri ali piši(A ali B), konstanta A, podatek (A ali B ali C) in konstanta B. Desno spodaj je izhod.

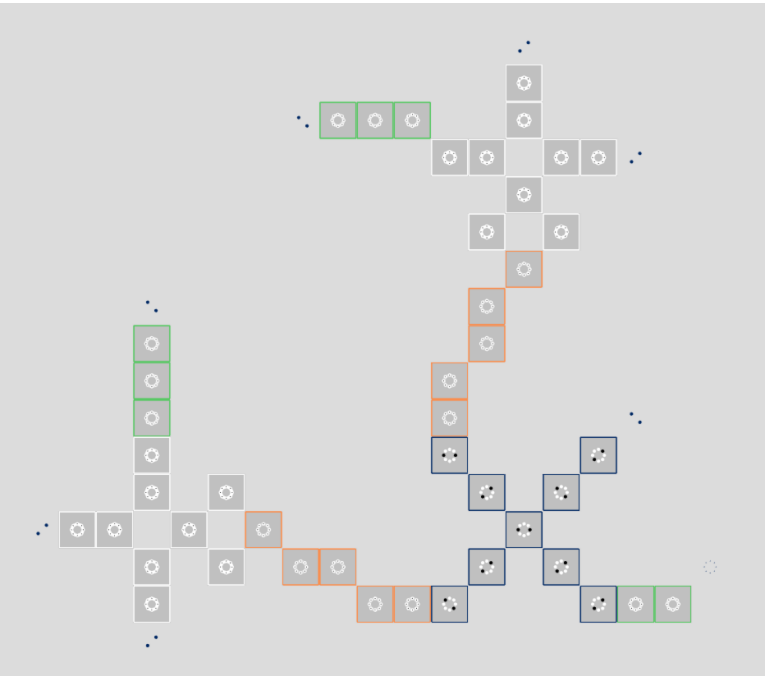
Karakteristična funkcija f^0

Ker karakteristična funkcija f^0, kot je ta predstavljena na sliki 12 potrebuje za pravilno delovanje vhode tako iz leve kot iz desne strani, in ker tega v USE shemi ne moremo doseči, smo jo poskusili implementirati na alternativen način. S tem smo želeli tudi preveriti, če je ideja o združevanju struktur s pomočjo USE sheme izvedljiva.

Spomnimo, da mora karakteristična funkcija f^0 vrniti vrednost B le za vhodno vrednost C, sicer pa vrne A. Torej jo lahko definiramo kot

$$f^0(x) \equiv \neg f^{-1}(x) \wedge \neg f^1(x).$$

Vse gradnike, potrebne za implementacijo takšne definicije pa imamo tako v USE shemi 5x5 kot tudi v 6x6. Slika 17 prikazuje implementacijo takšne strukture v shemi 5x5.



Slika 17. Karakteristična funkcija f^0 v USE 5x5 shemi. Na sliki je izpuščeno cepljenje vhodne vrednosti, ki se zgodi ali na sredini strukture, ali pa levo zgoraj.

Žal pa stvari ne delujejo, kot bi pričakovali. Kljub temu, da so posamezne komponente, ko smo jih testirali eno po eno, delovale popolnoma pravilno, je uporaba že samo dveh komponent pri vseh testiranih arhitekturah ter razdaljah med celicami privedla do napačnega obnašanja. Tako tudi celotna struktura ni vračala pravilnih vrednosti. Sumimo, da je tu problem predvsem v presluhu, oz. v tem, da celice kljub veliki oddaljenosti druga od druge nezanemarljivo vplivajo na preostale celice v strukturi.

Implikacija

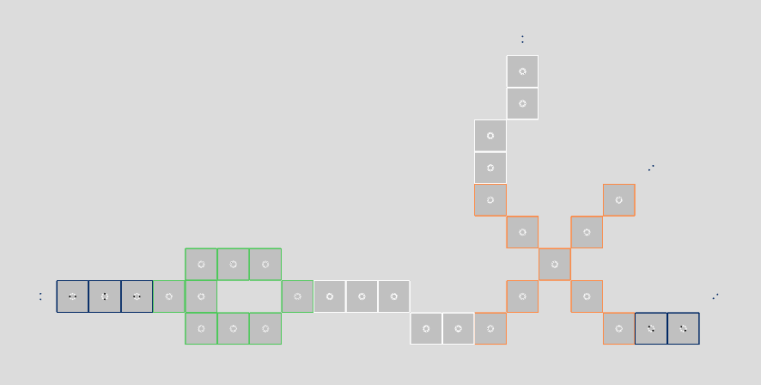
Kot demonstracijo združevanja struktur v shemi USE smo implementirali še implikacijo po Kleenejevi logiki. Implikacijo lahko preprosto zapišemo z logično enačbo

$$x \to y \equiv \neg x \vee y.$$

Struktura implikacije je predstavljena na sliki 18, pričakovani izhodi po Kleenejevi logiki pa v tabeli 10.

Tabela 10. Vhodi in želeni izhodi implikacije. Zadnji vhod B je fiksen in predstavlja vhod majoritetnim vratom, ki so v strukturi implikacije nastavljena na disjunkcijo.

vhod	izhod	vhod	izhod	vhod	izhod
A A B	B	B A B	A	C A B	C
A B B	B	B B B	B	C B B	B
A C B	B	B C B	C	C C B	C



Slika 18. Implikacija v USE 5x5 shemi. Sestavljena je iz negatorja in majoritetnih vrat.

Implikacija nam ni delovala kot smo si želeli, pravilnemu delovanju smo se najbolj približali pri nekaterih razdaljah v arhitekturah 60 (za razdalje $[170, 175]$, in 195) in 72 (za razdalje $[190, 200]$ ter 220). Pri teh kombinacijah implikacija na izhod vrne pravilen rezultat za vse vhode, razen za vhoda 'C A B' (izhod D namesto C) ter 'C C B' (izhod B namesto C).

Razlogov za neuspeh je najbrž več, zagotovo pa je eden od teh dejstvo, da za implementacijo implikacije nismo mogli uporabiti originalnih majoritetnih vrat, kot so predstavljena na sliki 14. Za potrebe zlaganja z ostalimi strukturami, smo morali vhode v majoritetna vrata pripeljati malce drugače, kar je delno pokvarilo že sama majoritetna vrata in posledično tudi implikacijo.

Zaključek

Pri prvem delu projekta smo bili precej uspešni. Kljub majhnim zapletom nam je uspelo implementirati osnovne strukture linij (diagonalna, ravna), njihovo križanje na več načinov, negator, majoritetna vrata (oblik '$\times$' in '+'), pomnilniško celico ter vse tri variante karakterističnih funkcij.

Več težav smo imeli pri drugem delu projekta. Težko je bilo strukture, ki za pravilno delovanje zahtevajo uporabo več urinih faz spremeniti tako, da bi delovale tudi v shemi USE. To nam je uspelo za linije, dve karakteristični funkciji in negator, pri drugih po so se pojavljali problemi. Na primer majoritetna vrata smo sicer uspeli spraviti v USE shemo vendar so delovala le za nekaj kombinacij arhitektur in razdalj. Ko smo jih želeli vključiti v večjo strukturo (implikacijo), so že ob najmanjši modifikaciji prenehala delovati pravilno. Prav tako so majoritetna vrata povzročala nekaj problemov pri implementaciji pomnilniške celice, a jih je bilo po drugi strani tudi lažje spraviti do delovanja, saj so morala pravilno delovati le za nekaj vhodov.

Implementacije osnovnih struktur v shemi USE, ki bi omogočile elegantno zlaganje in združevanje osnovnih struktur v večje zapletenejše strukture so zahteven problem, ki nam ga v tem projektu ni uspelo rešiti v celoti. Smo pa nekaj obstoječih delujočih struktur iz literature testirali za različne parametre in različne kombinacije ter poiskali in predstavili spodnje in zgornje meje za njihovo delovanje. Upamo, da bo naše poročilo vsaj malo prispevalo k nadaljnem razvoju tehnologije tQCA.

Literatura

1. Lent, C. S., Tougaw, P. D., Porod, W. & Bernstein, G. H. Quantum cellular automata. *Nanotechnology* **4**, 49 (1993).
2. Mraz, M. & Lebar Bajec, I. Večstanjsko procesiranje v strukturah kvantnih celičnih avtomatov. *Elektrotehniški vestnik* **2/3**, 105–110 (2006).
3. Campos, C. A. T., Marciano, A. L., Neto, O. P. V. & Torres, F. S. Use: A universal, scalable, and efficient clocking scheme for qca. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **35**, 513–517 (2016).
4. Magdevski, Z. & Mraz, M. *Realizacija omejenega nabora trovrednostnih logičnih funkcij na osnovi struktur kvantnih celularnih avtomatov: magistrska naloga* (UL-FRI, Ljubljana, 2006).
5. Mraz, M. *et al.* Towards multistate nanocomputing: The implementation of a primitive fuzzy controller. In *Second International Conference on Quantum, Nano and Micro Technologies (ICQNM 2008)*.
6. Janež, M. *et al.* Automatic design of optimal logic circuits based on ternary quantum-dot cellular automata. *WSEAS Trans. Cir. and Sys.* **7**, 919–928 (2008).
7. Pečar, P., Mraz, M. & Lebar Bajec, I. *Uporaba adiabatnega pristopa pri realizaciji trojiškega procesiranja na osnovi kvantnih celičnih avtomatov: magistrska naloga* (UL-FRI, Ljubljana, 2007).
8. Pečar, P., Ramšak, A., Zimic, N., Mraz, M. & Lebar Bajec, I. Adiabatic pipelining: a key to ternary computing with quantum dots. *Nanotechnology* **19**, 495401 (2008).
9. Pečar, P., Mraz, M., Zimic, N., Janež, M. & Lebar Bajec, I. Solving the ternary quantum-dot cellular automata logic gate problem by means of adiabatic switching. *Japanese Journal of Applied Physics* **47**, 5000 (2008).
10. Pečar, P., Janež, M., Zimic, N., Mraz, M. & Lebar Bajec, I. The ternary quantum-dot cellular automata memorizing cell. In *2009 IEEE Computer Society Annual Symposium on VLSI*, 223–228 (IEEE, 2009).
11. Pečar, P. & Lebar Bajec, I. The key elements of logic design in ternary quantum-dot cellular automata. In *International Conference on Unconventional Computation*, 177–188 (Springer, 2011).
12. Jeraj, G., Levičnik, U., Mujanović, Z. & Skube, S. Implikacija v tQCA (2009).
13. Peršolja, M., Finžgar, L., Dimitrieski, V. & Simčič, M. Seminarska naloga: Enotritni komparator v tQCA (2009).
14. Erlih, T., Ilc, N., Tišler, M. & Vidonja, D. Seminar pri predmetu ONT: tQCA 1x2 RAM (2008).
15. Struna, S., Andrejak, L., Jež, J. & Turšič, M. Analiza večnivojske gradnje trojiških QCA (tQCA) struktur (2010).
16. Božič, D., Krajačič, I., Cesar, A. & Sakelšak, D. Testiranje delovanja struktur realiziranih s trojiškimi kvantnimi celičnimi automati (2011).
17. Lampreht, B., Stepančič, L., Vizec, I. & Žankar, B. tQCA - Seštevalnik (2008).
18. Rolih, M., Lebar Bajec, I. & Mraz, M. *Analiza možnosti realizacije logičnih reverzibilnih vrat v trostanjskem kvantnem celičnem avtomatu: diplomska naloga* (UL-FRI, Ljubljana, 2013).
19. Lebar Bajec, I. & Pečar, P. Two-layer synchronized ternary quantum-dot cellular automata wire crossings. *Nanoscale research letters* **7**, 1 (2012).

Doprinosi avtorjev

T.M. je predelal polovico pregleda literature, pognal simulacije križanja linij in pomnilniške celice. Dodal je križanje linij v ravnini, znova testiral pomnilniško celico in sorodna dela uredil bolj kronološko. Popravil sorodna dela, literaturo, tabele pri križanju celic in pomnilniška celica. Opisal je USE shemo in sestavil USE pomnilniško celico. Popravil je navadno pomnilniško celico in tudi USE pomn. celico, dodal je tudi začetek uvoda.
A.D. je predelal drugo polovico literature, pognal simulator nad horizontalno in diagonalno linijo ter negatorju, ter si ogledal majoritetna vrata. Nato je popravil prvo verzijo negatorja, implementiral majoritetna vrata ter dodal še vertikalno linijo ter linijo preko slojev. Vse strukture je tudi pognal in s pomočjo bisekcije odkril intervale pravilnega delovanja, spisal del uvoda ter dopolnil opise preostalih struktur v USE shemi, ter poskušal v isti shemi realizirati še majoritetna vrata oblike '+'. Za konec

je poskusil še implementirati karakteristično funkcijo f^0 v USE strukturi.
R.I. je preučil dosedanje delo, prebral izbrano literaturo ter se seznal s simulatorjem in skonstruiral ter testiral tri karakteristične funkcije. Nato je preveril delovanje ter v USE shemi implementiral majoritetna vrata oblike ‘x’ in implikacijo.