Seminarska naloga: Enotritni komparator v tQCA

UNI RS 4.letnik- Optične in nanotehnologije

Ljubljana, 10.1. 2009

Matej Peršolja

Luka Finžgar

Vlado Dimitrieski

Matej Simčič

1.)UVOD

1.1) Cilji

Naša naloga je bila realizacija enotritnega komparatorja s celicami trojiških kvantnih celičnih avtomatov. Odločitev za trojiško logiko izhaja iz znanih dejstev o ekonomičnosti take vrste zapisa[3]. Pomen komparatorja se bistveno spreminja ob prehodu iz dvojiške v trojiško logiko. Torej, komparator v dvojiški logiki pomeni 1 če sta bita enaka, sicer pa ničla(lahko tudi obratno). Bistvo je da informacija, ki jo bitni komparator da na izhod ena med dvemi vrednostmi, sta ali nista vhoda enaka. V trojiški logiki imamo več možnosti za izvedbo komparatorja, ker posredujemo več informacije. In sicer, ker ima trit po definiciji 3 vrednosti, poleg informacije o enakosti oziroma neenakosti lahko zagotovimo informacijo o tem, ali je prvi vhod večji oziroma manjši od drugega(seveda, lahko tudi obratno). To izvedbo oziroma to funkcijo v [1] so poimenovali "Magnitude function y" z naslednjo tabelo:

x2\x1	0	1/2	1
0	1/2	1	1
1/2	0	1/2	1
1	0	0	1/2

Naslednja možnost, ki zadošča definiciji kompartorja in nam vrača isto informacijo v primerjavo s prvo rešitvijo, le da je zapis drugačen je, je že definirana funkcija[1] Exclusive Max. Se pravi, če sta vhoda enaka dobimo 0, sicer dobimo večjega med vhodi. Pravilnostna tabela te funkcije:

x2\x1	0	1/2	1
0	0	1/2	1
1/2	1/2	0	1
1	1	1	0

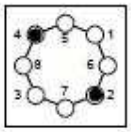
Zadnja med izbranimi možnostmi je preprosta preslikava funkcije komparatorja iz dvojiške v trojiško logiko. Ko sta vhoda enaka imamo eno vrednost, recimo 0, sicer ostale dve. Pravilnostna tabela:

x2\x1	0	1/2	1
0	1	!=1	!=1
1/2	!=1	1	!=1
1	!=1	!=1	1

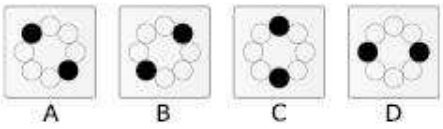
Zadnjo rešitev bomo poskusili implementirati straight forward. Enacbe so enake kot pri ekvivalenci:X1*X2 + notX1*notX2.
Lahko bi poskusili se implementirat zadnjo rešitev s pomočjo seštevalnika oziroma odštevalnika[6].

2.2) Osnove tQCA in primitivi

Lentovo dvojiško qca celico(bQCA) razširimo tako, da sedaj vsebuje 8 kvantnih pik s krožno porazdelitvijo.

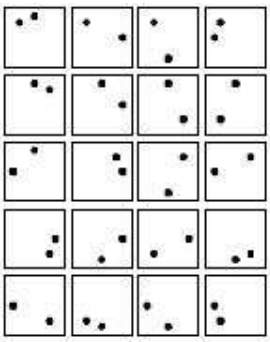


Elektrona v celicah tunelirata med pikami zaradi odbojnih sil med njima ali pa zaradi interakcije z okoljem. Elektrona v taki celici se postavita v 4 pozicije, ki ustrezajo največji oddaljenosti med elektronoma in s tem najmanjši odbojni sili (fizikalno ozadje je predstavljeno v [4],naš model pa se opira na polprevodniško QCA implementacijo s GaAs).



Osnovna stanja trojiške kvantne celice

Možna so še druge razporeditve elektronov, vendar ta stanja niso stabilna:



Štiri pozicije ustrezajo stanjem A,B,C in D. Stanji A in B (elektrona v diagonali) sta enaki kot pri binarni QCA(logična 0 in 1). Vertikalno lego označimo s C in horizontalno z D.

Po dogovoru se vrednosti D odpovemo kot vrednosti na vhodni celicah. Na vhodih in izhodih pričakujemo le vrednosti{A,B,C} kar ustreza trovrednostnemu sistemu(0,1,1/2), vrednost D pa dopuščamo le kot vrednost v "notranjih" celicah strukture.

Podobno kot v QCA imamo tudi v tQCA več tipov celic:
• vhodne celice: v njih se pod vplivom zunanjih sil prisilno doseže želena razporeditev elektronov
• notranje celice: v njih se izvede poljubna logična 'obdelava' vhodnih podatkov

• izhodne celice: njihovo stanje (s časovnim zamikom) obravnavamo kot rezultat procesiranja

Ker so vhodi in izhodi na robovih strukture, vmes pa se izvaja procesiranje na notranjih celicah, gre pri tQCA tako kot pri bQCA za robno gnano procesiranje.

Pri tQCA lahko uporabimo nekatere primitivne, ki so nam znani iz bQCA. Negator in linija delujeta pravilno, pri čemer moramo pri prenosu vrednosti C in D po liniji paziti na liho dolžino. Če uporabimo 4-fazni urin cikel za sinhronizacijo prenosa podatkov delujejo tudi kotna linija, razvejitvena linija(fanout) ter klasična majoritetna vrata.

Pravilnostna tabela komparatorja:

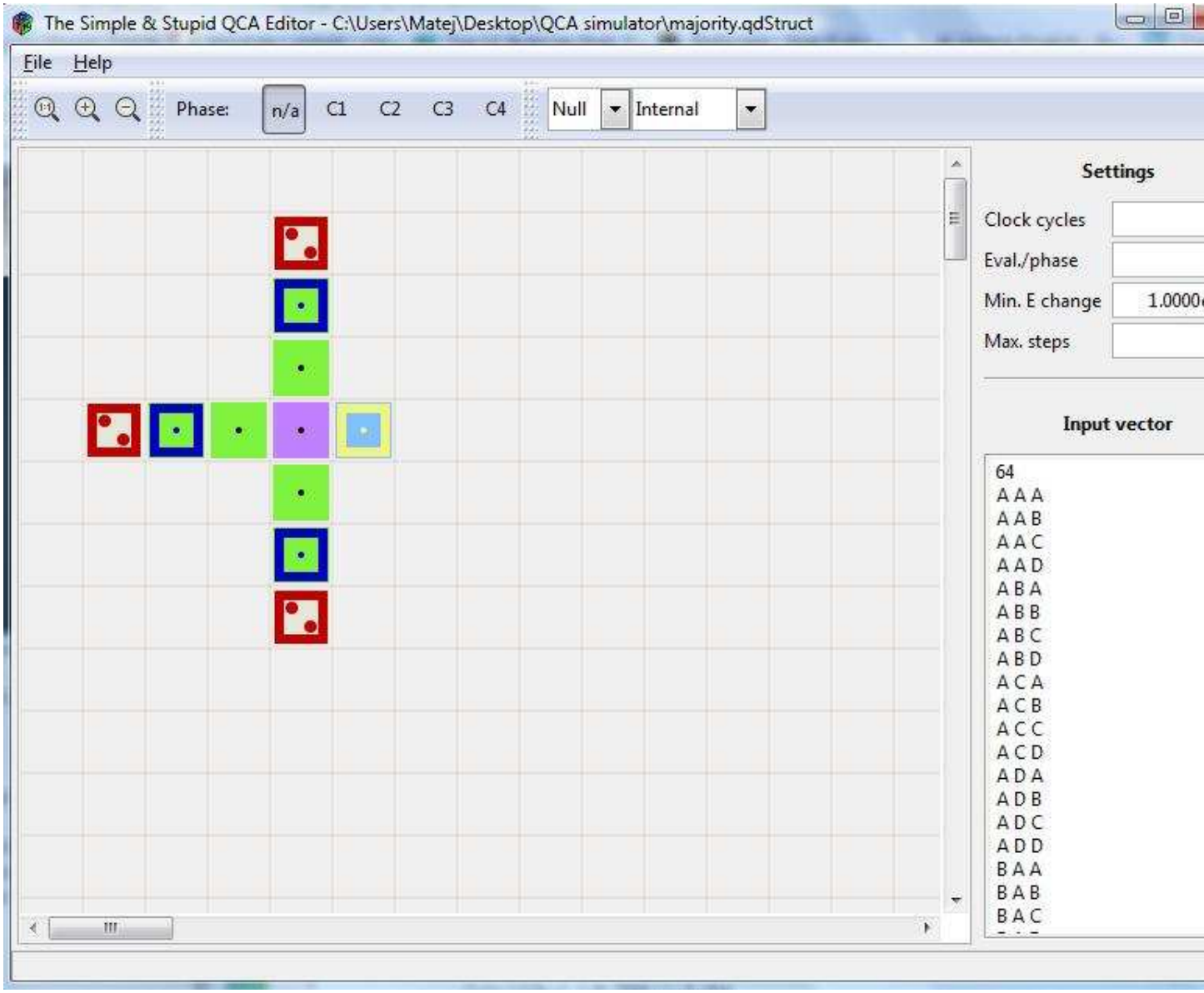
0	0	1
0	1	0
0	1/2	0
1	0	1/2
1	1	1
1	1/2	1/2
1/2	0	1/2
1/2	1	0
1/2	1/2	1

A	A	B
A	B	A
A	C	A
B	A	C
B	B	B
B	C	A
C	A	C
C	B	C
C	C	B

2.)METODE

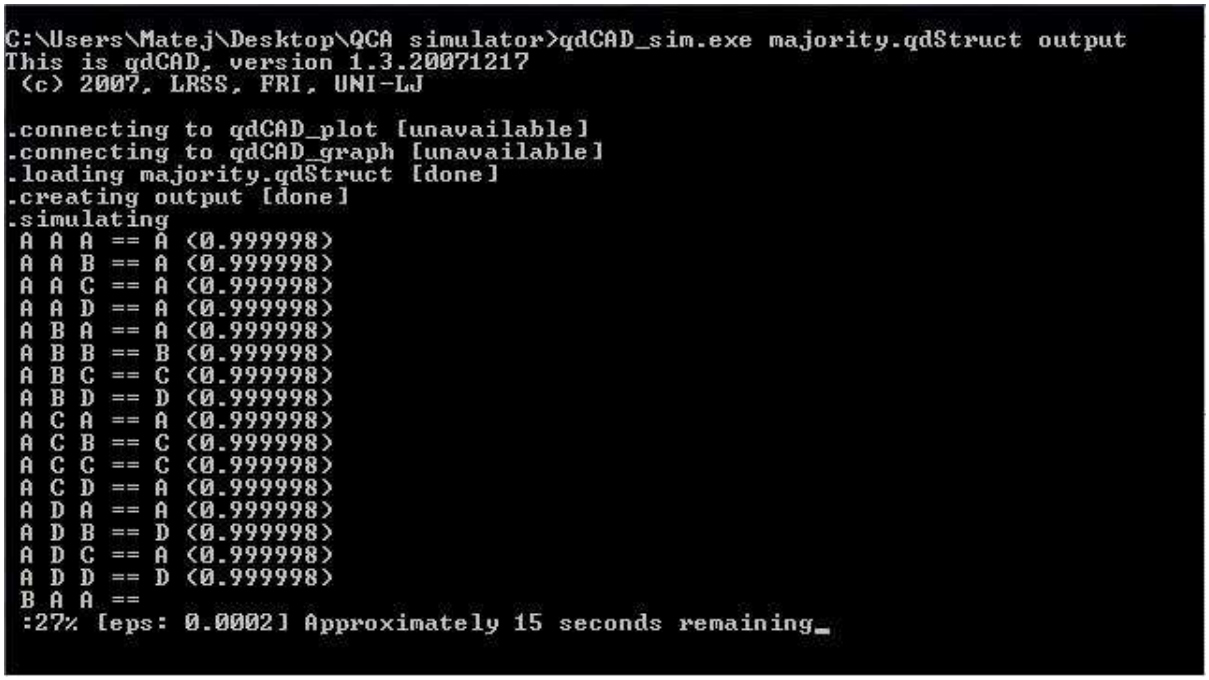
Postopek testiranja:

1. V programu QCAEdit(razvit v laboratoriju za računalniške strukture in sisteme na Fakulteti za računalništvo in informatiko) narišemo strukturo. Komponentam določimo clocke in definiramo nek vhodni vektor. Delo shranimo v datoteko, ki ji damo končnico qdstruct.



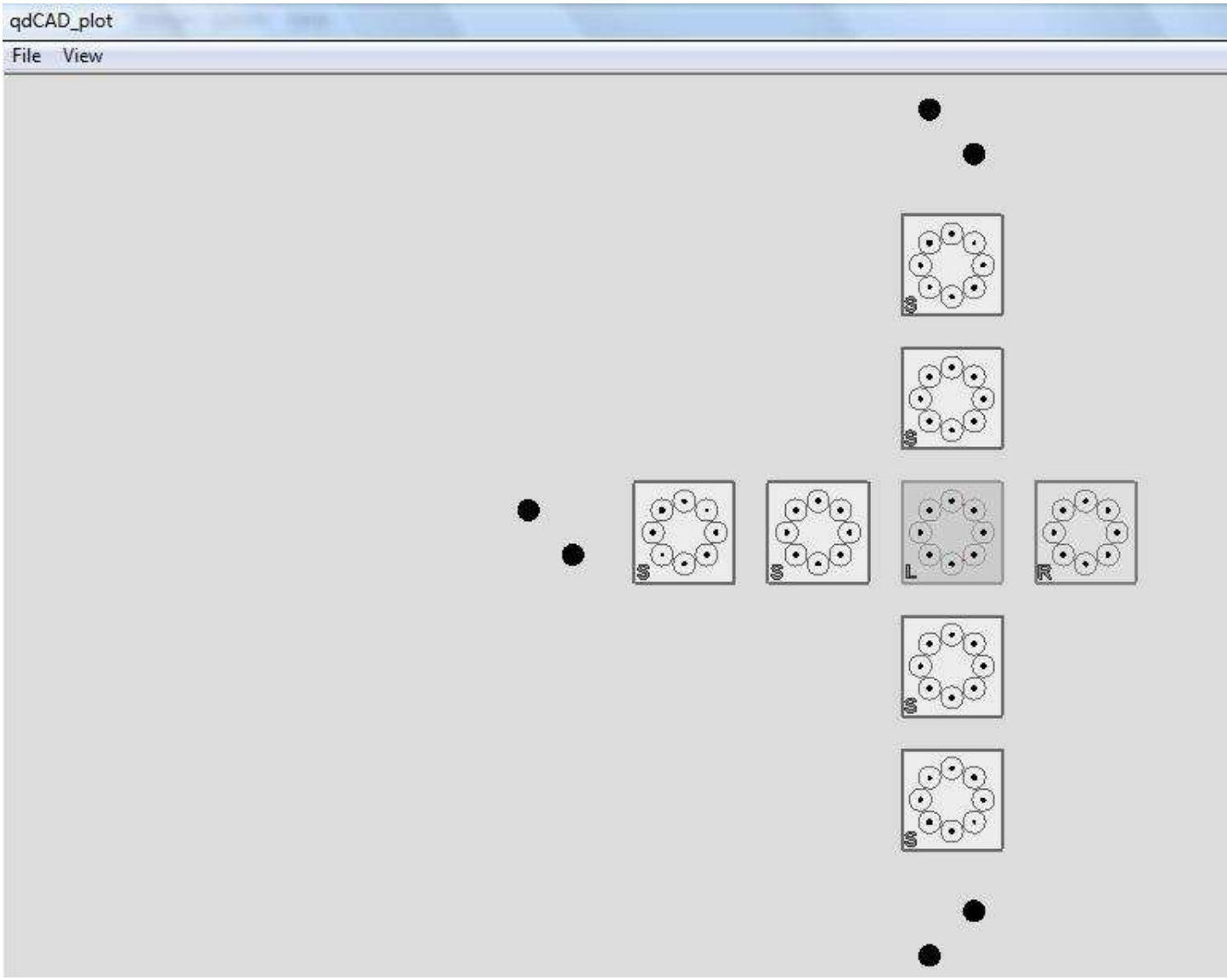
Risanje strukture v programu qca editor

2. V konzoli poženemo simulacijo s pomočjo programa qdCAD_sim, ki mu kot vhodni parameter podamo vhodno datoteko (tisto, ki jo bomo testirali) in izhodno datoteko, kamor se shranijo podatki o testiranju.
3. Med simuliranjem se v konzolo izpisujejo rezultati testiranja. To so vhodne in izhodne vrednosti. Podrobnejši podatki - stanja ostalih celic, se zapisujejo v izhodno datoteko.



V konzoli pognano testiranje datoteke majority.qdStruct. Rezultati se bodo zapisali v datoteko output.

4. Izhodno datoteko po končani simulaciji odpremo v programu qdCAD_plot(razvit v laboratoriju za računalniške strukture in sisteme na Fakulteti za računalništvo in informatiko), kjer se nam izriše struktura. Strukturo lahko simuliramo po korakih in na ta način vizualno spremljamo, kako je potekala simulacija, kar nam konzola in izhodna datoteka ne omogočata.



qdCAD_plot prikaz za output datoteko

Reševanje problema

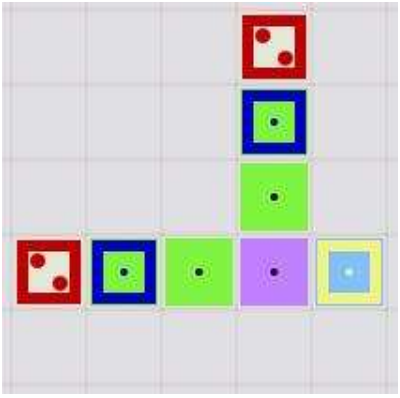
Po branju poročila lanske skupine z isto nalogo smo se odločili, da se problema ne bomo lotili s postavitvijo logičnih enačb in minimizacije. Preverili bomo čim več postavitev celic v prostoru 3x3 ter pogledali rezultate simulacije, če kakšna izmed postavitev vodi k rezultatu.

Na prostoru 3x3 smo naključno povezali nekaj celic in nato simulirali to vezje. Ko se je nabralo veliko podatkov smo preverili, če kakšna kakšna postavitev predstavlja rešitev ali vsaj vodi k rešitvi. Ker na 3x3 nismo imeli uspeha, smo poizkusili še nekaj postavitev na večjem 4x4 prostoru.

3.)REZULTATI

a)(prostor 3x3)

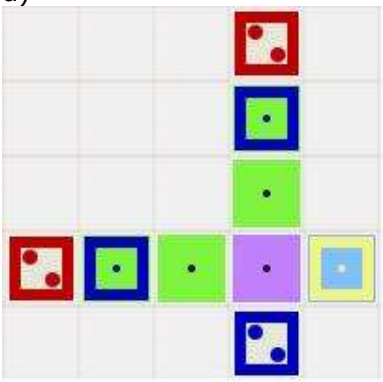
1. Struktura:



Rezultati:
A A == A (0.999998)
A B == C (0.999998)
A C == C (0.999998)
B A == D (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

Opis:

2. Struktura:
a)



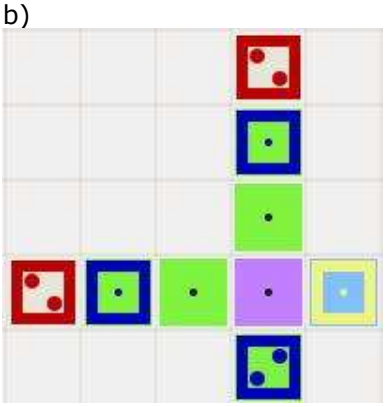
Rezultati:

A A == A (0.999998)
A B == C (0.999998)

A C == C (0.999998)
B A == D (0.999998)
B B == B (0.999998)
B C == B (0.999998)
C A == A (0.999998)
C B == C (0.999998)
C C == C (0.999998)

Opis:

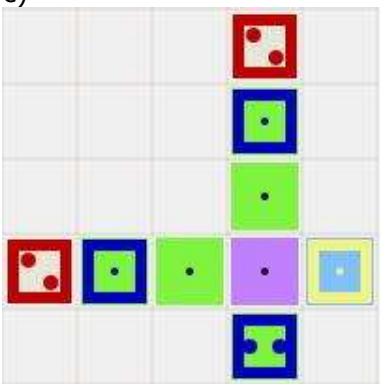
b)



Rezultati:

A A == A (0.999998)
A B == D (0.999998)
A C == A (0.999998)
B A == C (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == B (0.999998)
C C == C (0.999998)

c)

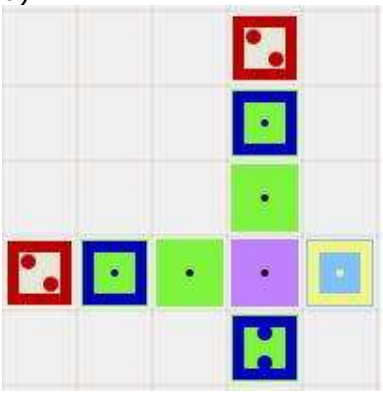


Rezultati:

A A == A (0.999998)
A B == D (0.999998)
A C == A (0.999998)
B A == C (0.999998)
B B == B (0.999998)
B C == C (0.999998)

C A == C (0.999998)
C B == B (0.999998)
C C == C (0.999998)

d)



Rezultati

A A == A (0.999998)
A B == D (0.999998)
A C == A (0.999998)
B A == C (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == B (0.999998)
C C == C (0.999998)

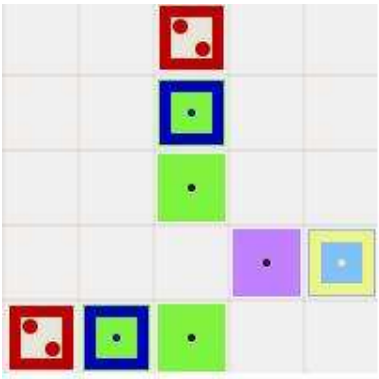
3. Struktura:

Rezultati:

A A == A (0.999998)
A B == A (0.999998)
A C == A (0.999998)
B A == B (0.999998)
B B == B (0.999998)
B C == B (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

Opis:

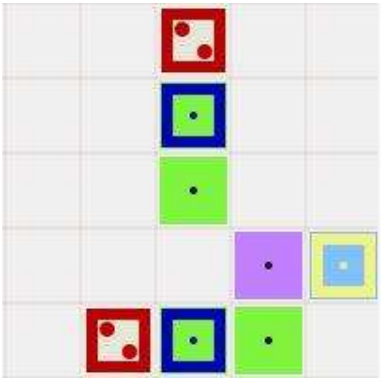
4. Struktura



Rezultati:

A A == B (0.999998)
A B == A (0.999998)
A C == B (0.999998)
B A == C (0.999998)
B B == A (0.999998)
B C == A (0.999998)
C A == B (0.999998)
C B == A (0.999998)
C C == D (0.999998)

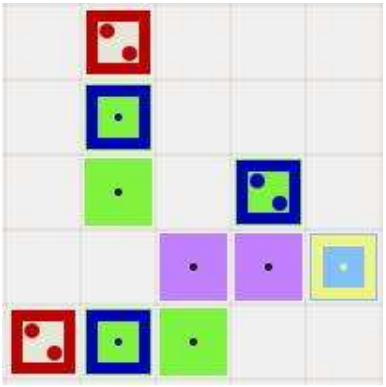
5. Struktura



Rezultati:
A A == A (0.999998)
A B == B (0.999998)
A C == C (0.999998)
B A == A (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == A (0.999998)
C B == B (0.999998)
C C == C (0.999998)

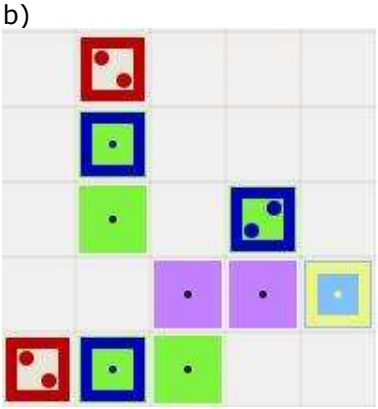
6. Struktura

a)



Rezultati:
A A == B (0.999998)
A B == B (0.999998)
A C == D (0.999998)
B A == A (0.999998)
B B == A (0.999998)
B C == D (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

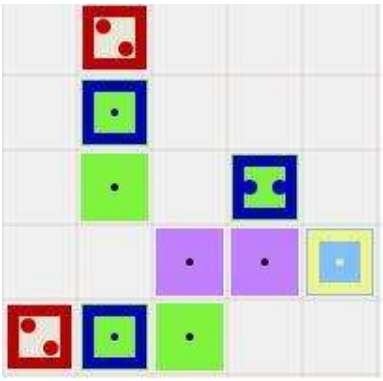
b)



Rezultati:

A A == B (0.999998)
A B == B (0.999998)
A C == D (0.999998)
B A == A (0.999998)
B B == A (0.999998)
B C == D (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

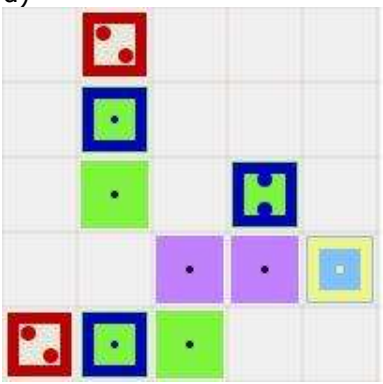
c)



Rezultati:

A A == N (0.823223)
A B == N (0.823223)
A C == N (0.823223)
B A == N (0.823223)
B B == N (0.823223)
B C == N (0.823223)
C A == N (0.823223)
C B == N (0.823223)
C C == N (0.823223)

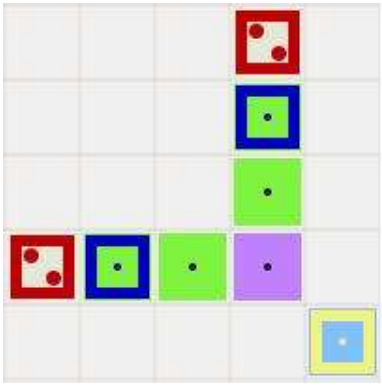
d)



Rezultati:

A A == B (0.999998)
A B == B (0.999998)
A C == D (0.999998)
B A == A (0.999998)
B B == A (0.999998)
B C == D (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

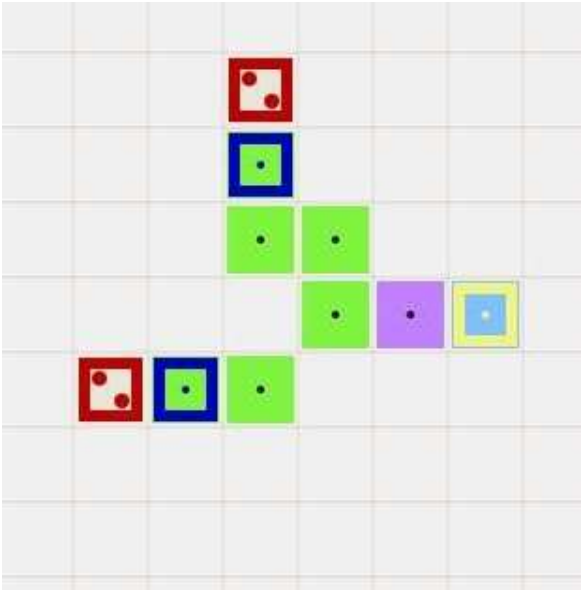
7. Struktura



Rezultati:

A A == B (0.999998)
A B == C (0.999998)
A C == D (0.999998)
B A == D (0.999998)
B B == A (0.999998)
B C == D (0.999998)
C A == D (0.999998)
C B == D (0.999998)
C C == D (0.999998)

8.struktura

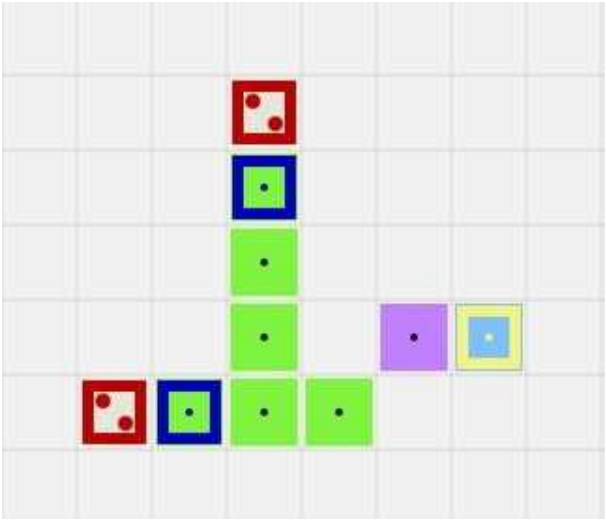


Rezultati:

A A == D (0.998687)
A B == D (0.998687)
A C == D (0.998687)
B A == D (0.998687)
B B == D (0.998687)
B C == D (0.998687)
C A == C (0.99851)
C B == C (0.99851)

C C == C (0.99851)

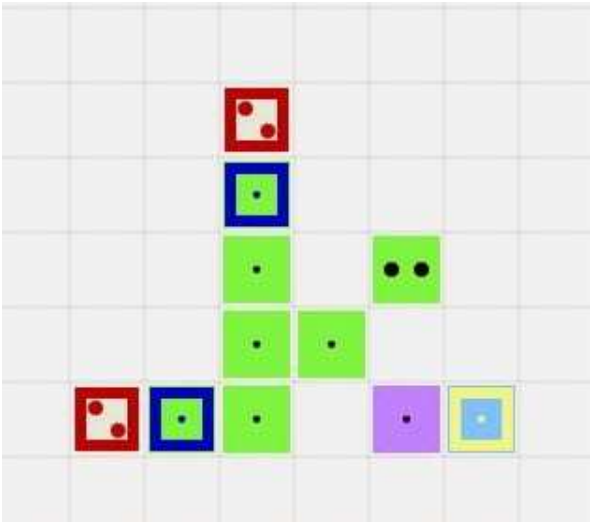
9.struktura



Rezultati:

A A == B (0.998585)
A B == C (0.998508)
A C == C (0.998507)
B A == C (0.998508)
B B == A (0.998584)
B C == C (0.998507)
C A == C (0.998508)
C B == C (0.998508)
C C == C (0.998508)

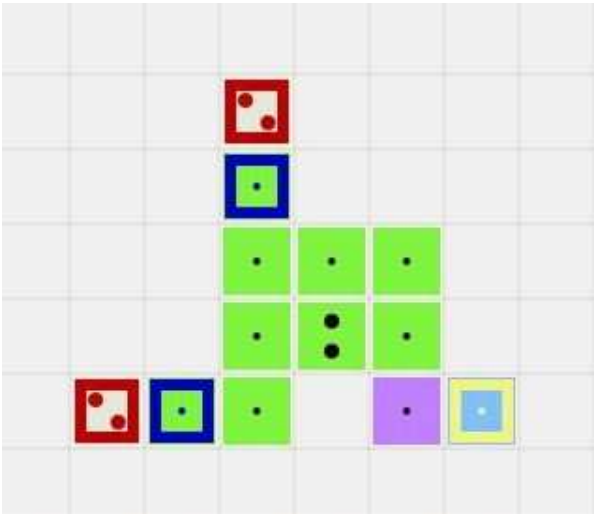
10. struktura



Rezultati:

A A == D (0.998686)
A B == D (0.998686)
A C == D (0.998686)
B A == D (0.998686)
B B == A (0.998585)
B C == D (0.998686)
C A == D (0.998686)
C B == D (0.998686)
C C == D (0.998686)

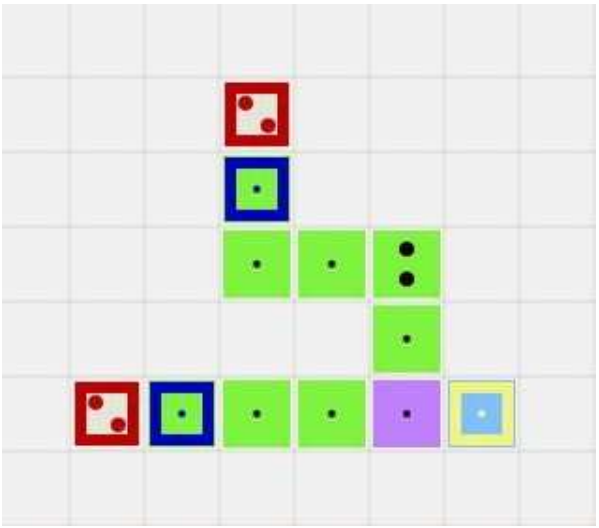
11.struktura



Rezultati:

A A == C (0.998505)
A B == C (0.998505)
A C == D (0.998688)
B A == C (0.998505)
B B == C (0.998505)
B C == D (0.998688)
C A == D (0.998688)
C B == D (0.998688)
C C == D (0.998688)

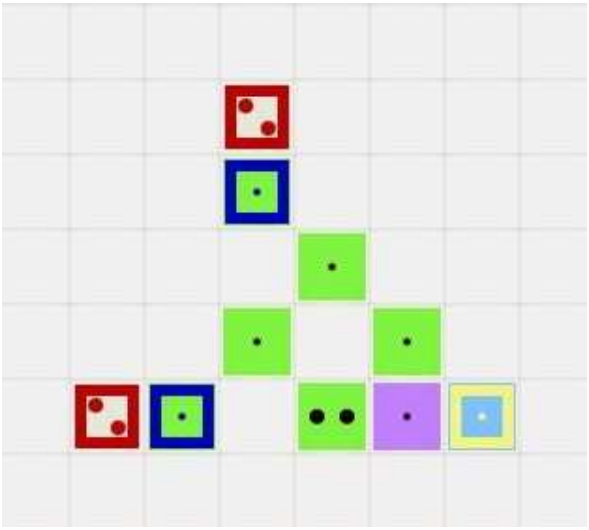
12.struktura



Rezultati:

A A == C (0.998508)
A B == C (0.998508)
A C == D (0.998688)
B A == C (0.998508)
B B == C (0.998508)
B C == D (0.998688)
C A == B (0.998585)
C B == B (0.998585)
C C == D (0.998688)

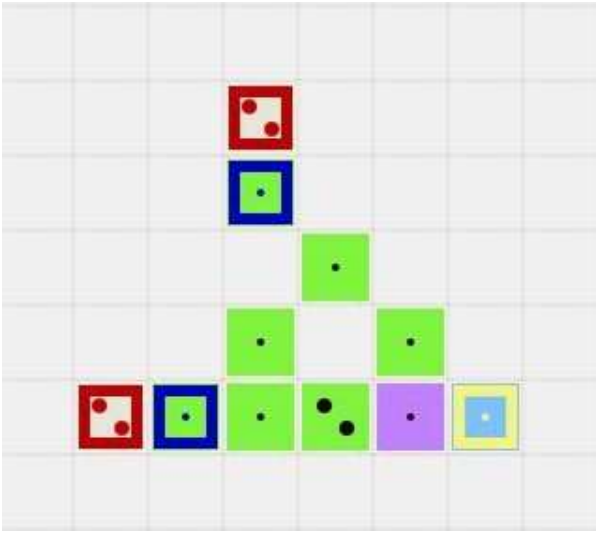
13.struktura



Rezultati:

A A == C (0.998508)
A B == C (0.998508)
A C == C (0.998507)
B A == D (0.998688)
B B == D (0.998688)
B C == D (0.998688)
C A == D (0.998688)
C B == C (0.998508)
C C == D (0.998688)

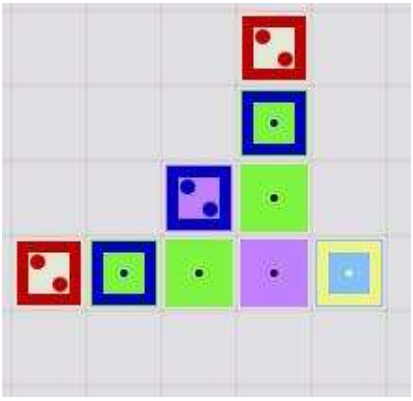
14.struktura



Rezultati:

A A == D (0.998688)
A B == D (0.998688)
A C == D (0.998688)
B A == D (0.998688)
B B == D (0.998688)
B C == D (0.998688)
C A == D (0.998688)
C B == D (0.998688)
C C == D (0.998688)

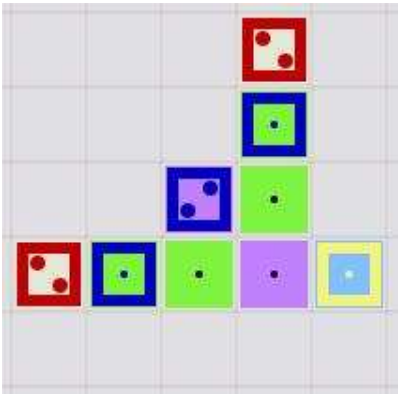
15.struktura



Rezultati:

A A == A (0.999998)
A B == C (0.999998)
A C == C (0.999998)
B A == D (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

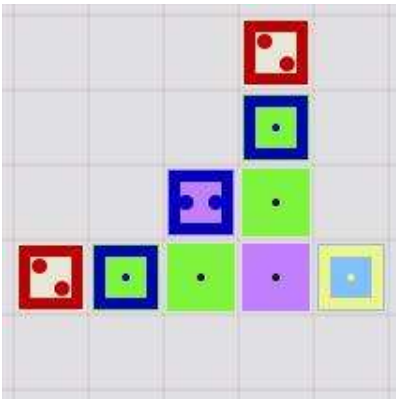
16.struktura



Rezultati:

A A == A (0.999998)
A B == C (0.999998)
A C == C (0.999998)
B A == D (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

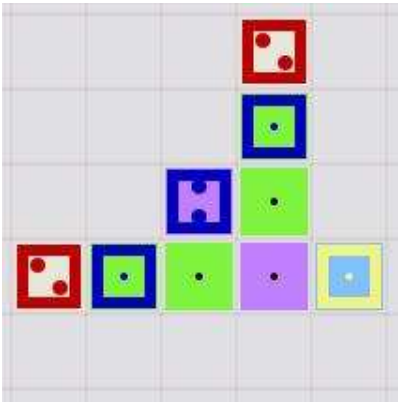
17. struktura



Rezultati:

A A == A (0.999998)
A B == C (0.999998)
A C == C (0.999998)
B A == D (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

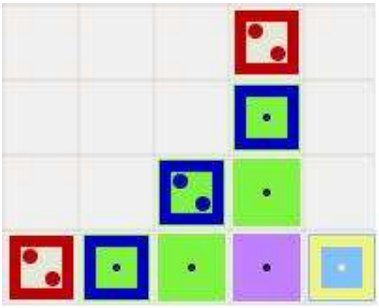
18.struktura



Rezultati:

A A == A (0.999998)
A B == C (0.999998)
A C == C (0.999998)
B A == D (0.999998)
B B == B (0.999998)
B C == C (0.999998)
C A == C (0.999998)
C B == C (0.999998)
C C == C (0.999998)

19. struktura

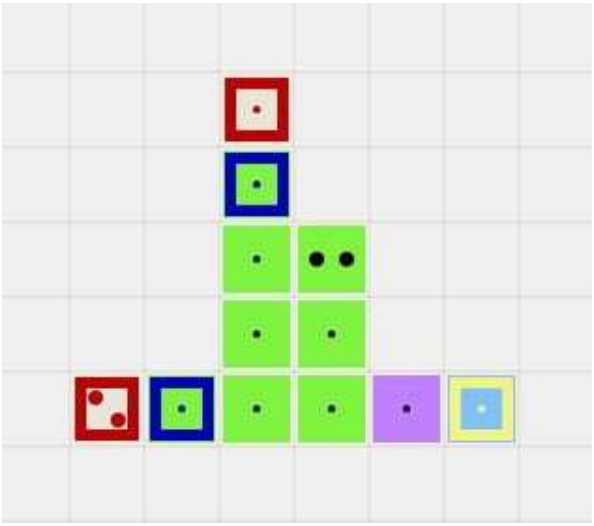


Rezultati:

A A == C (0.999998)
A B == C (0.999998)
A C == C (0.999998)
B A == D (0.999998)
B B == D (0.999998)
B C == C (0.999998)
C A == C (0.999998)

C B == C (0.999998)
C C == C (0.999998)

20.struktura

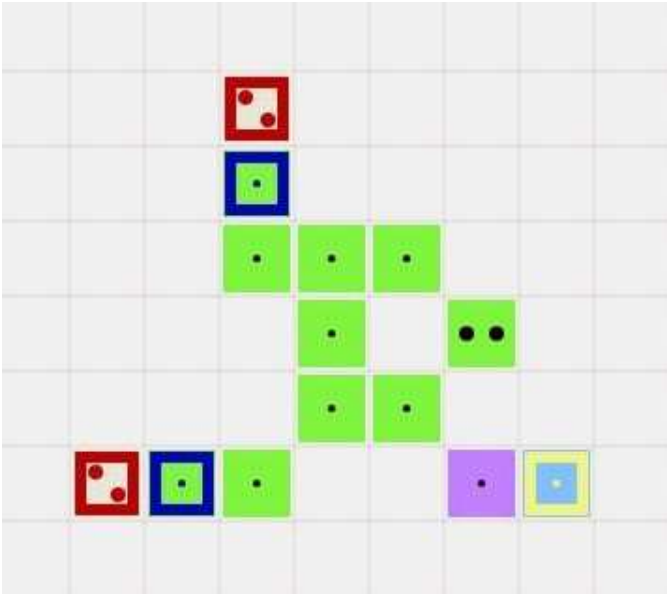


Rezultati:

A A == C (0.99851)
A B == D (0.998687)
A C == D (0.998687)
B A == D (0.998687)
B B == D (0.998687)
B C == D (0.998687)
C A == D (0.998687)
C B == D (0.998687)
C C == D (0.998687)

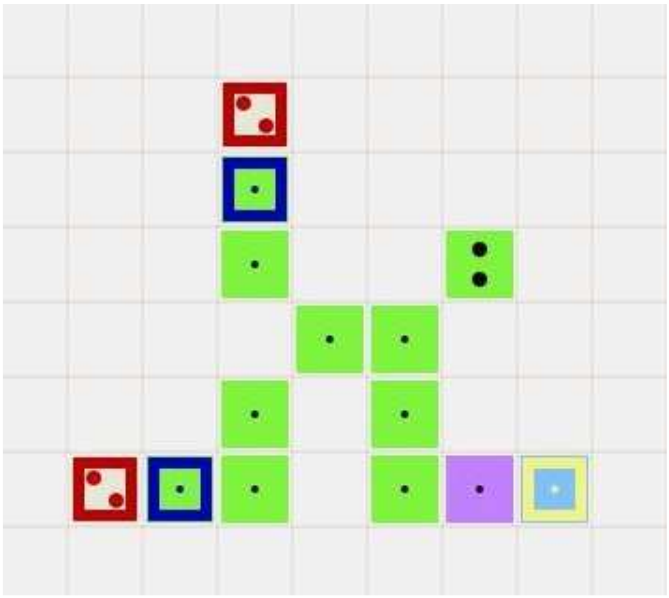
b)prostor 4x4

1.struktura



Rezultati A A == C (0.998508)
A B == C (0.998508)
A C == C (0.998508)
B A == C (0.998508)
B B == C (0.998508)
B C == C (0.998508)
C A == D (0.998687)
C B == D (0.998687)
C C == D (0.998686)

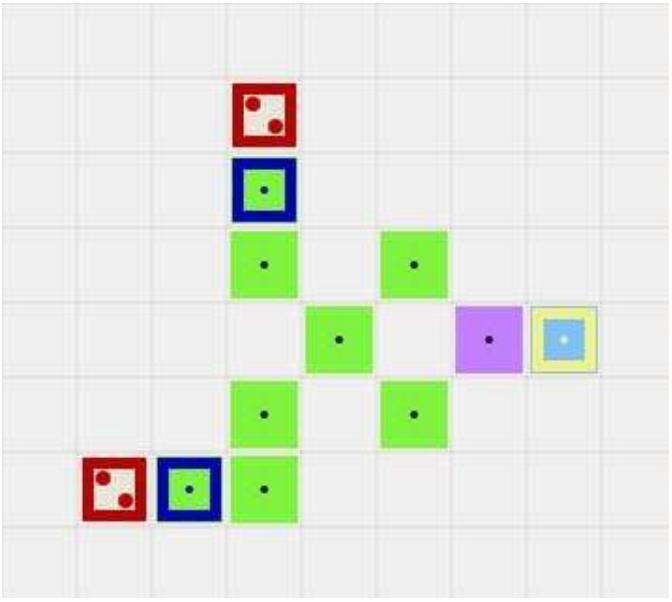
2.struktura



Rezultati:

A A == C (0.99851)
A B == C (0.99851)
A C == C (0.99851)
B A == D (0.998687)
B B == D (0.998687)
B C == C (0.99851)
C A == D (0.998687)
C B == D (0.998687)
C C == D (0.998687)

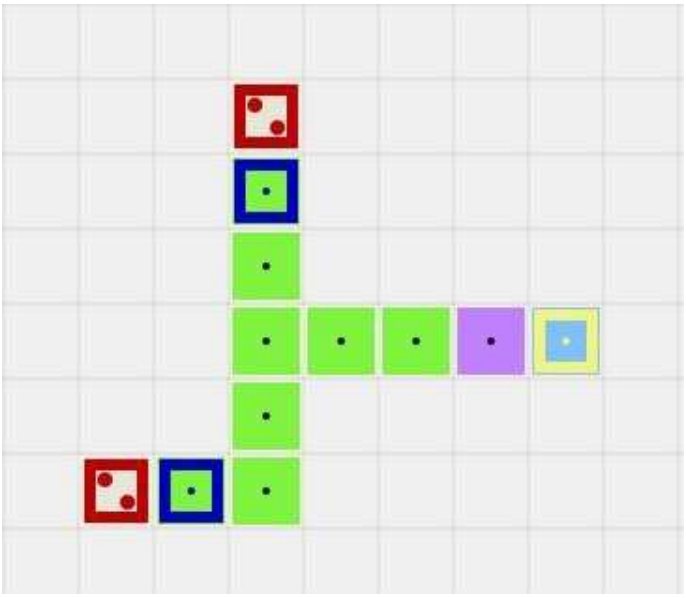
3.struktura



Rezultati:

A A == B (0.998585)
A B == B (0.998585)
A C == B (0.998585)
B A == B (0.998585)
B B == A (0.998585)
B C == A (0.998585)
C A == B (0.998585)
C B == A (0.998585)
C C == A (0.998585)

4.struktura

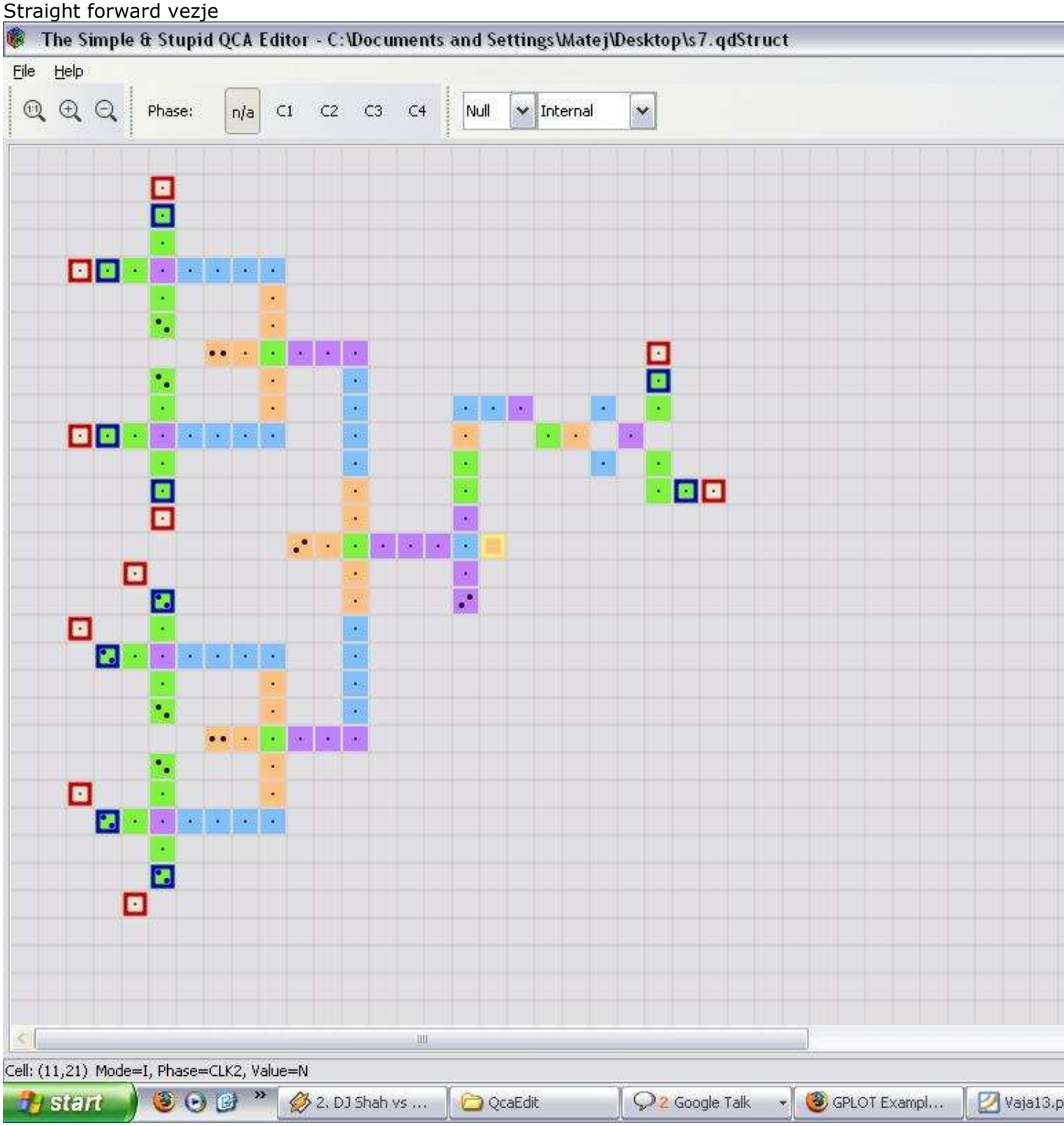


Rezultati:

A A == C (0.99851)
A B == C (0.99851)
A C == C (0.99851)
B A == C (0.99851)

B B == C (0.99851)
B C == C (0.99851)
C A == D (0.998687)
C B == D (0.998687)
C C == D (0.998687)

Straight forward vezje



Rezultati: N/A

3.1) DISKUSIJA

Pri postavljanju naključnih struktur v prostoru velikosti 3x3 smo dobili zelo slabe rezultate,med katerimi so bili izhodi "približno" podobni željnim le pri strukturah 4,9(ujemanja pri štirih vhodih). Obetali smo si rezultatov, kjer bi bila le 1 ali 2 vrednosti napačni in bi ti vrednosti lahko filtrirali s katerimi izmed že znanih tQCA struktur. Dokaj gotovi smo, da se komparatorja na tako majhnem prostoru(ali vsaj dela) ne da narediti.
Nato smo se usmerili na prostor 4x4, kjer pa smo kmalu obupali,saj je možno zelo veliko različnih postavitev.
Verjetno bi bilo 4x4 prostor možno hitreje kot z risanjem preiskati s programom, ki bi preizkusil vse možne postavitve(na način ki ga je izbrala druga skupina s tQCA).

Težave pri risanju vezij v QcaEdit in simulaciji

Ko smo skušali narediti svoje strukture smo dobili vedno vse izhode vrednost N, zato smo se poglobili v samo delovanje in odkrili, da ko shranimo neko strukturo, ta pri spremenljivki "maximal distance of influence" privzame vrednost 10,kar pa ne omogoča uspešne simulacije.
Če ta parameter popravimo v tekstovni datoteki na 3000, kot je to v testnem primeru majority.qdStrudct, so se strukture pravilno simulirale. Ročno popravljanje po vsakem risanju vezja je bilo zelo zamudno, zato bi bilo dobro spremeniti program.
Urejevalnik bi bilo dobro spremeniti tudi tako,da bi se ob risanju vezja nove celice takoj prikazale brez klika na gumb za ponastavitev zuma. To smo tudi poizkusili(saj je na voljo izvorna koda), a smo imeli težave pri prevajanju programa zaradi manjkajočih knjižnic.

Priročno bi bilo tudi, da bi obstajala kratka navodila za uporabo simulatorja. Za prvo silo smo si pomagali s komentarji v datoteki majority.qdStruct. Pozorni moramo tudi biti, kadar hočemo doloceno vrednost implicirati na celice; vedno more biti za celico Electrode celica tipa Driver.

4.)ZAKLJUČEK

Naše iskanje komparatorja v 3x3 strukturi(in nekaj 4x4) žal ni prineslo sadov, upamo pa da bo to poročilo v pomoč ostalim pri nadaljnjem iskanju. Mogoče smo bili preveč naivni, da nam bo uspelo na

roke s popravki priti do rešitve, a vseeno se nam je zdelo bolj smiselno, kot pa da bi se naloge lotili na klasični način in na koncu prišli do vezja z velikim številom celi, ki pa ga v simulatorju ne bi mogli preizkusiti ter bi tako delal le na papirju. Samo testiranje nam je vzelo dosti časa,

tako da bi predlagali, da bi postavili nek GRID, ki bi bil dostopen študentom za testiranje podobnih problemov(predvsem za avtomatsko testiranje vseh možnih kombinacij postavitev).

Za konec bi še pohvalil orodje Google Docs v katerem smo skupaj delali na poročilu in prezentaciji. Dosti nam je olajšalo skupinsko delo.

5.)VIRI

[1] http://www.trinary.cc/

[2] I.LebarBajec, M.Mraz: Večstanjsko procesiranje v strukturah kvantnih celičnih avtomatov. Elektroteh. vestn., 2006, letn. 73, no. 2/3, str. [105-110]

[3] Hayes, B., 2001. Third base. American Scientist, vol. 89, num. 6, pp. 490-494.

[4] I.LebarBajec, M. Mraz,N. Zimic,P. Pečar,A. Ramšar: Adiabatic pipelining:a key to ternary computing with quantum dots

[5].Lebar Bajec, I., Mraz, M., 2006. Večstanjsko procesiranje v strukturah kvantnih celičnih avtomatov. Elektrotehniški vestnik, vol. 73, num. 2-3, 105-110. (Trojiški QCA)

[6] Lampreht, B., Žankar, B., Stepančič, L., Vizec, I., 2007. *tQCA - Seštevalnik*. seminarska naloga, Ljubljana

[7] prof. Miha Mraz: predloge za predavanja pri predmetu ONT