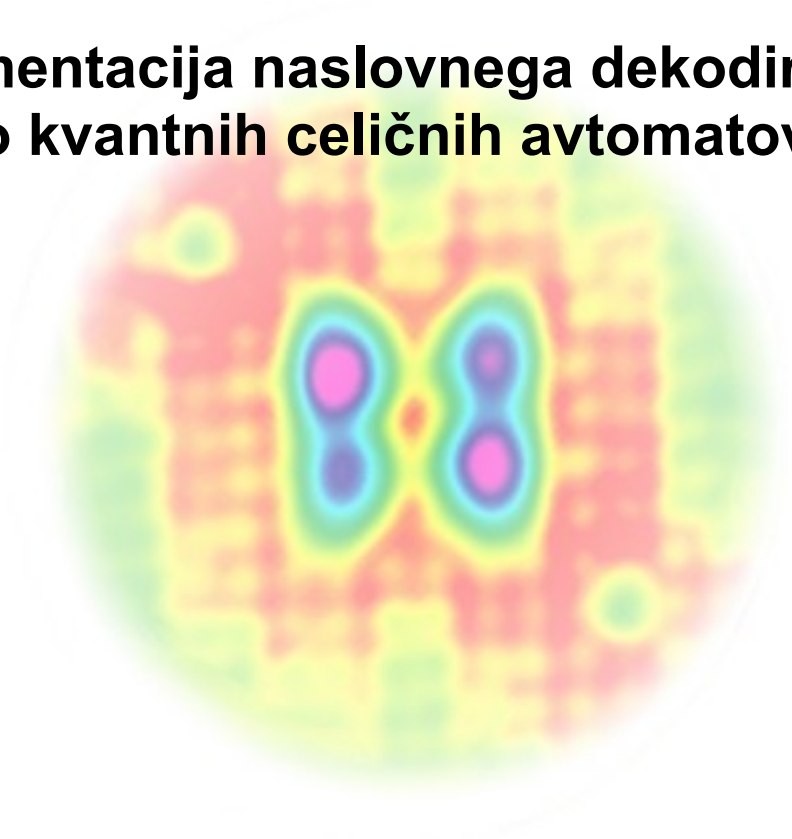
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Optične in nanotehnologije

Implementacija naslovnega dekodirnika z uporabo kvantnih celičnih avtomatov (QCA)



Skupina 1:

Darko Božić, Ivan Krajačić, Anže Cesar, Dejan Sakelšak

Ljubljana, 30.11.2010

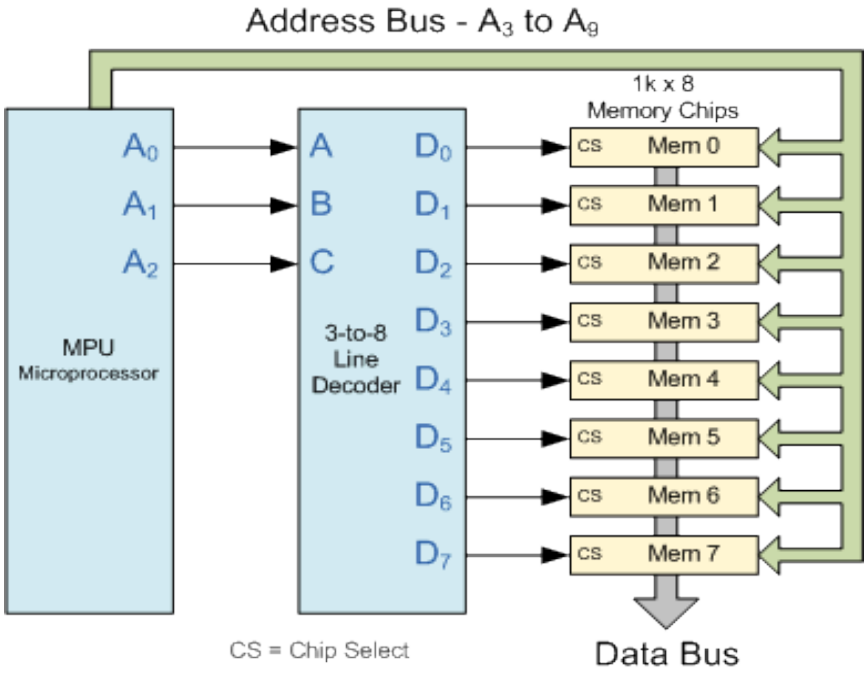
Kazalo vsebine

1. Uvod

Naša naloga pri prvi seminarski nalogi je bila implementirati trobitni naslovni dekodirnik z dvobitnim podatkovnim vodilom. Zaradi potrebe po testiranju delovanja dekodirnika, je bilo potrebno implementirati tudi majhen (8x2bit) pomnilnik iz JK celic. Za pravilno delovanje je bilo poleg naslovnih signalov in podatkovnih signalov potrebno dodati še signal za določanje smeri toka podatkov (R/W signal), s katerim izbiramo ali gre za pisanje v pomnilnik ali branje iz njega. Za lažjo realizacijo pa smo privzeli, da sta vhod in izhod ločena (in tako ne predstavljata ravno vodila), saje je implementacija trostanjskih izravnalnikov s QCA celicami zgodba zase.

2.Naslovni Dekodirnik

Pomnilnik je sestavljen iz pomnilniških besed, v katere z raznimi enotami pišemo in iz njih beremo. Da bi lahko besede med seboj razlikovali, jim določimo enolične številke katerim pravimo naslov. Ob dostopu do neke pomnilniške besede moramo torej pomnilniku povedati naslov besede s katero želimo delati. Ker so pomnilniki realizirani z različnimi tehnologijami, imajo tudi različne načine naslavljanja posameznih besed.

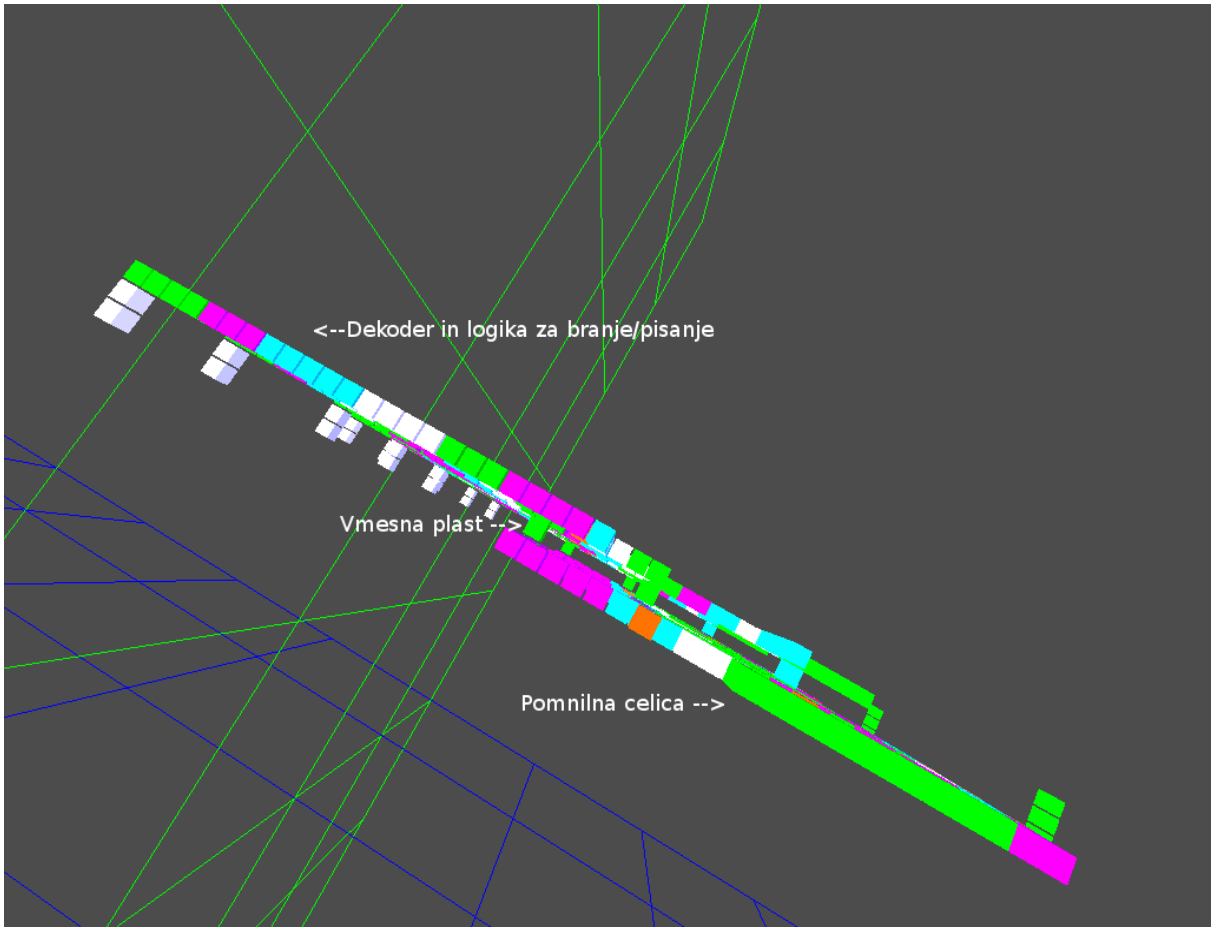


Slika 1: Primer naslovnega dekodirnika

Pri DRAM-u so to vrstice in stolpci, pri SRAM-u so to signali, ki povzročijo branje ali pisanje podatka na ali iz pomnilnih celic na oziroma iz podatkovnega vodila. Pri vhodno/izhodnih sistemih, pa preko krmilnikov naprav s signali za omogočitev naprav napravi povemo, naj z vodila vzame podatek ali naj ga nanj zapiše. Vsa ta pretvarjanja med naslovom in signali, ki so specifični za posamezno okolje, opravljajo naslovni dekodirniki. V našem primeru gre za dekodirnik, ki iz naslova razbere kateri par celic bo vključil bodisi za branje bodisi za pisanje.

3.Opis Vezja

Vezje se razteza skupno čez 32 plasti. Vsaka pomnilniška celica se nahaja na svoji plasti in sicer celici, ki se odzivata na naslov A=000 se nahajata na isti plasti, celici ki se odzivata na naslov A=001 na svoji itd. Tako imamo skupaj 8 plasti z dvema pomnilnima celicama. Poleg plasti s samimi celicami, vsakemu paru pomnilnih celic pripada tudi plast na kateri se nahaja naslovni dekodirnik za dani pomnilni celici in ustrezni signali RW, A0, A1, A2, Din1 in Din2. Tako je celotni naslovni dekodirnik razdeljen po celem vezju za čim bolj pravilno delovanje vezja. Na omenjeni plasti je še dodana logika za pravilno branje in pisanje v celico. Zaradi pravilnosti delovanja je med vsako pomnilno celico in njeno pripadajočo logiko tudi mejna plast.



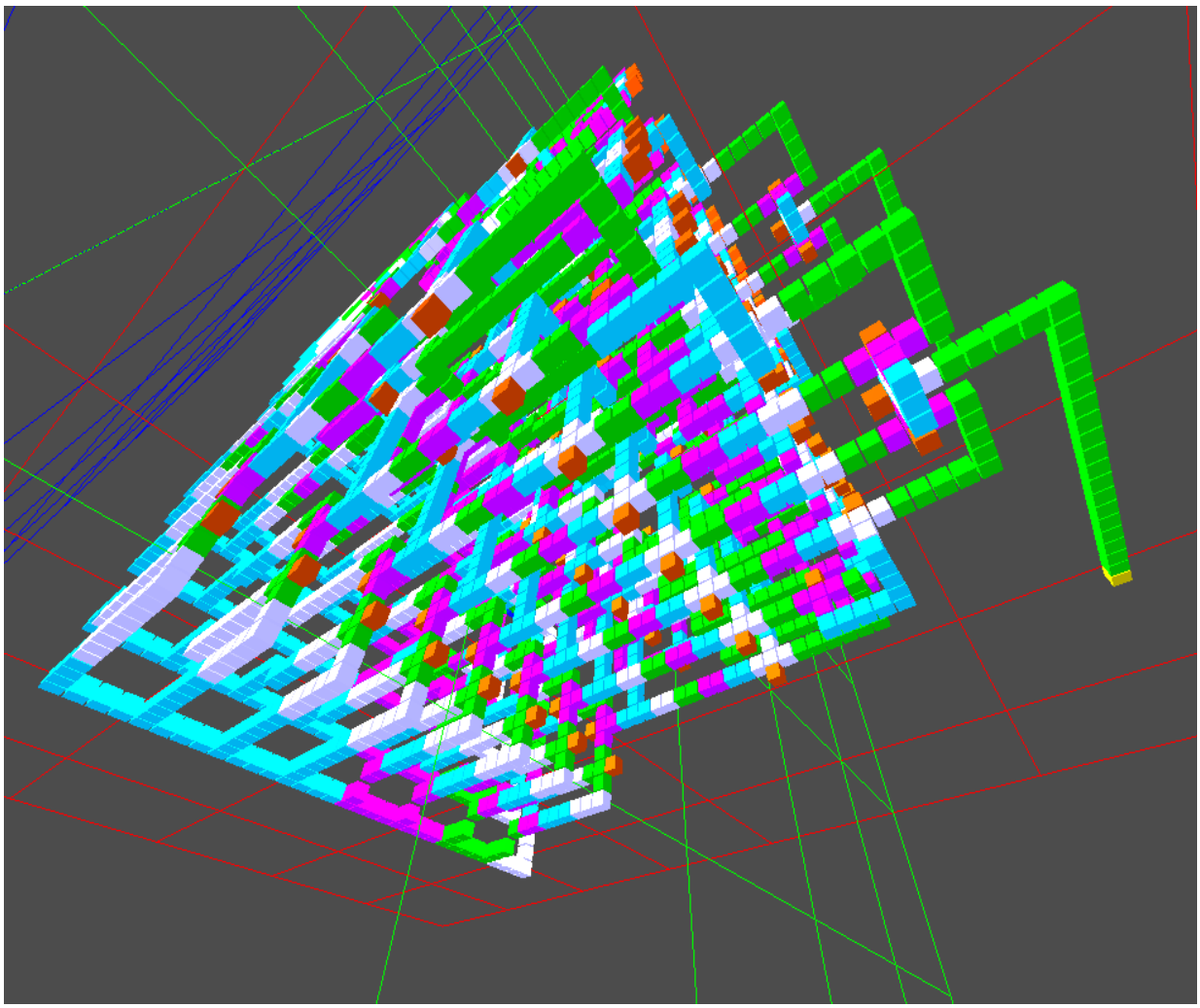
Slika 2: Pogled na plasti

Tako nam osnovo za gradnjo predstavlja 3 plastna arhitektura:

- 1. plast: pomnilna celica
- 2. plast: obvezna plast za prenos signala
- 3. plast: logika za branje in pisanje ter naslovni dekodirnik

4.3D Pogled na dekodirnik

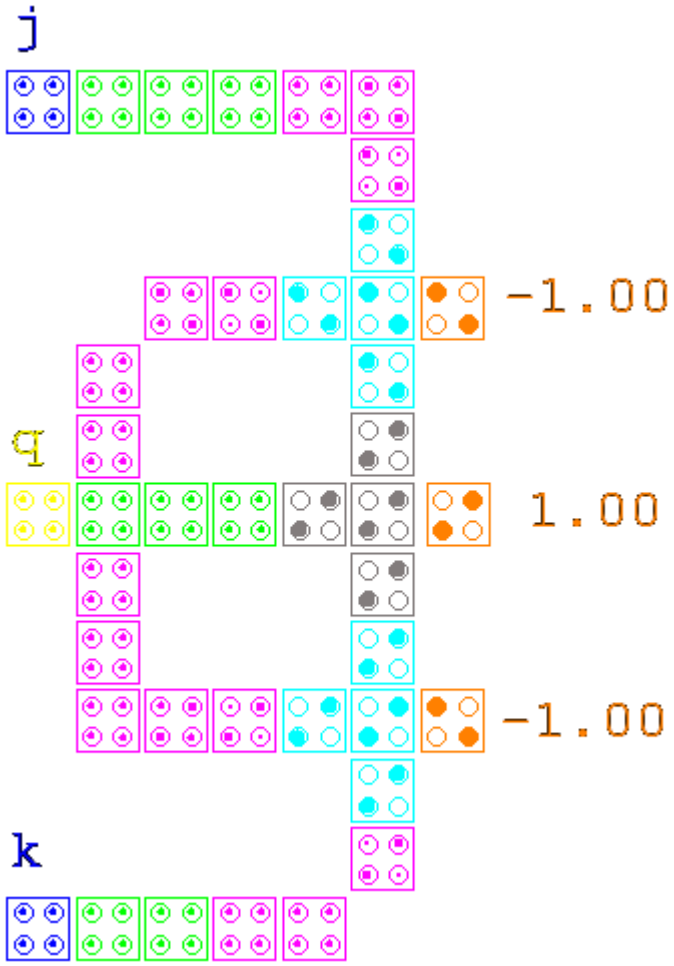
Ker je program QCA Designer odprto-koden program, smo imeli možnost popraviti izvorno kodo, da je izpisovala datoteke po naši želji. S pomočjo posebej napisanega programa, smo te datoteke prebrali in kvantne celice z uporabo OpenGL grafičnega programskega vmesnika prikazali v 3D prostoru, kar prikazuje slika 3.



Slika 3: 3d pogled na dekodirnik

5.Osnova pomnilne celice

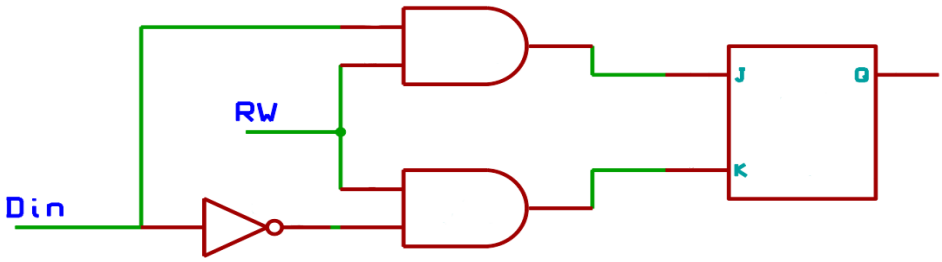
Podatek, ki se ga zapiše na določen naslov, se hrani v JK celici. Na spodnji sliki vidimo JK celico realizirano v QCA. Celica je zelo kompaktna in simetrična, kar nam zagotavlja dobro in stabilno delovanje. S signaloma J in K se nastavi stanje celice na 0 ali 1, ki se odraža tudi na izhodu Q ampak šele ob naslednji urini periodi.



Slika 4: Zgled implementacije JK celice

6.Pomnilna celica

Pomnilna celica deluje po logiki, ki je prikazana na sliki 5 po pravilih, ki jih podaja logična tabela za



Slika 5: Logika za vodenje JK pomnilne celice

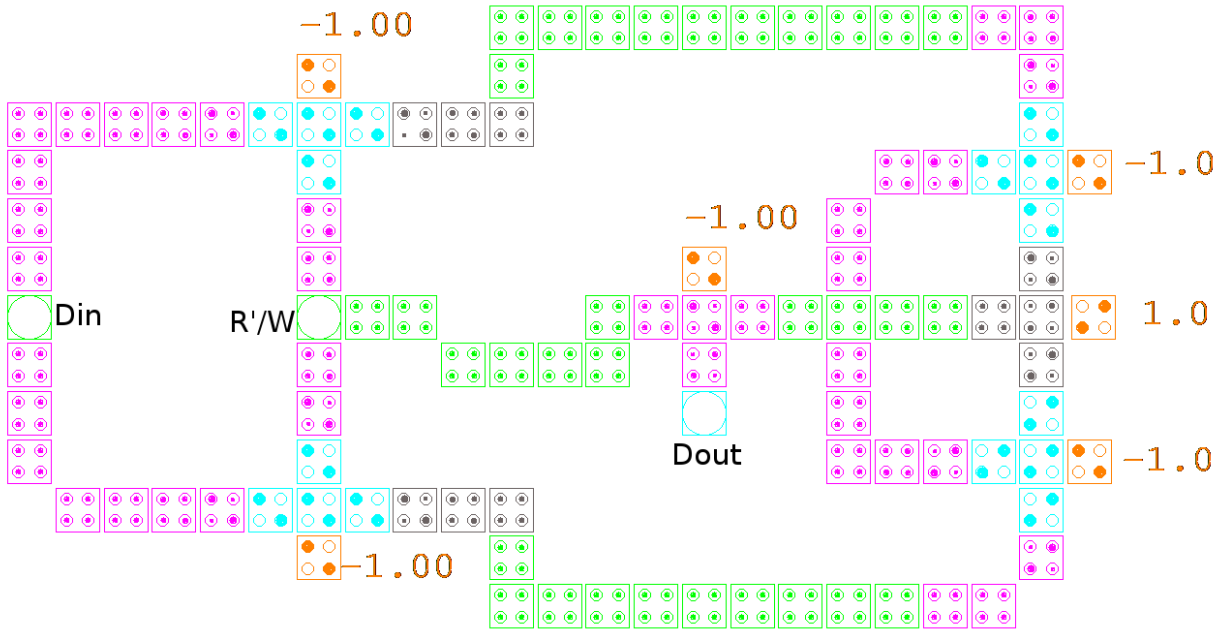
JK celico v tabeli 1. Iz tabele 1 je razvidno, da branje celice (RW=0), ne spreminja internega stanja celice. Kot posledica vrednosti signala RW=0 sta signala J in K enaka 0. Pisanje neke vrednosti v

Din	RW	J	K	Q
0	0	0	0	Q
0	1	0	1	0
1	0	0	0	Q
1	1	1	0	1

Tabela 1: Logična tabela delovanja JK celice

celico (ko je stanje signala RW=1) povzroči nastavitev signalov J in K, ki sta odvisna od signala Din, na takšno kombinacijo, da se v celico zapiše želena vrednost. Logika s slike 5 se tako preslika v QCA logiko na sliki 6.

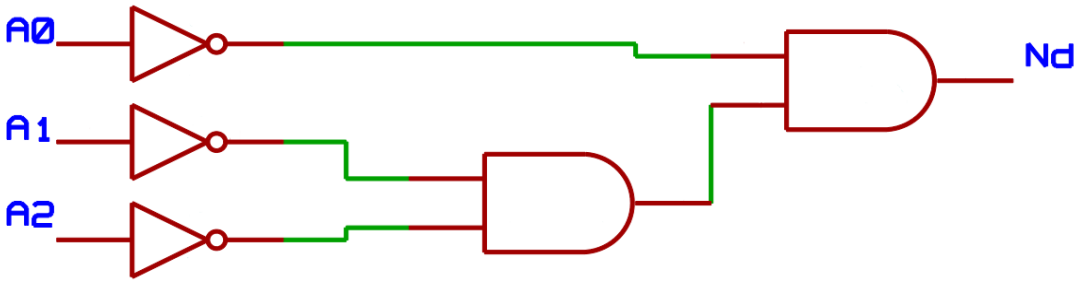
Na sliki 6 so vidna tudi dodatna majoritetna vrata (vrata roza barve pri Dout), ki so povezana na izhod JK celice. Ta vrata nam omogočajo prenos informacije iz celice na podatkovno vodilo v primeru, da smo sprožili branje na dani celici. V primeru, ko se informacija zapisuje v celico dobimo na izhodu Dout logično 0. Za vpis informacije potrebuje celica 2 urini periodi medtem ko branje iz celice poteka hitreje. Informacija je na voljo v Clock-u 2 v isti urini periodi, ko nastavimo ustrezna signala RW in Din. Zaradi preglednejše realizacije se vhoda RW, Din in izhod Dout, priključi preko nasledje plasti.



Slika 6: Implementirana struktura iz slike 5 z uporabo QCA

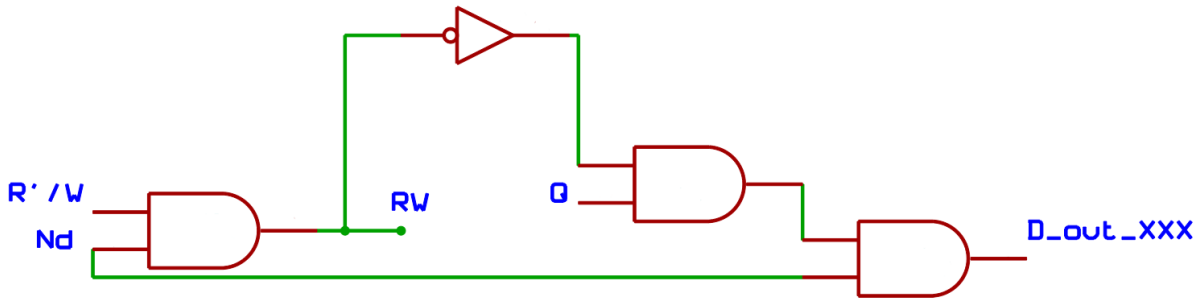
7. Dekodirnik in smerna logika

Slika 7 prikazuje del naslovnega dekodirnika za dekodiranje naslova A=000. Izhod Nd na na sliki 7 je enak 1 samo v primeru, ko so vsi signali Ai enaki 0. Za dekodiranje ostalih naslovov moramo primerno nastaviti operatorje na naslovnih vhodih A0, A1 in A2. Vsaka celica, ki se naslavlja ima svojo kombinacijo vhodnih operatorjev za prepoznavanje naslova.



Slika 7: Shema naslovnega dekodirnika za naslov A=000

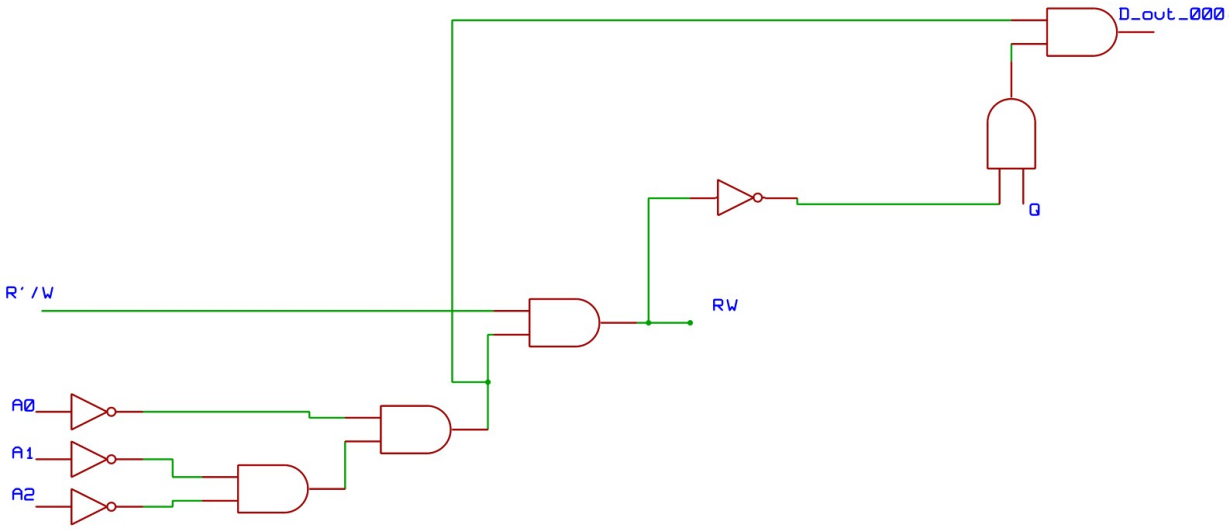
Slika 8 prikazuje bralno-pisalno logiko, ki skrbi za usmerjanje signalov. Izhod iz celice D_out_XXX je odvisen od signalov Nd in R'/W. V primeru Nd=0 se smatra, da celica ni bila naslovljena, zato je izhod D_out_XXX=0. V primeru Nd=1, pa se informacija Q prenese na izhod D_out_XXX v primeru branja (R'/W=0). V primeru pisanja (R'/W=1) pa se na izhod prenese logična 0.



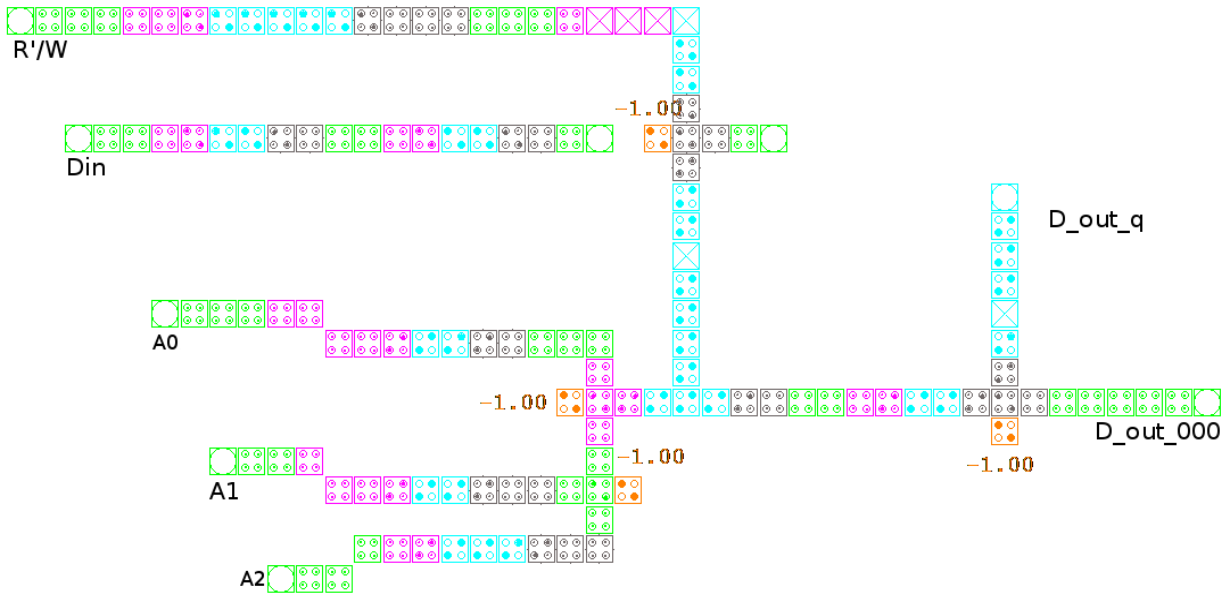
Slika 8: Shema bralno-pisalne logike

V primeru branja, bi ustrezen podatek iz celice dobili v 3 urini periodi. Signal D_out_q je izhod iz JK pomnilne celice.

Na sliki 9 je logično vezje dekodirnika z bralno-pisalno logiko na sliki 10 pa QCA implementacija.



Slika 9: Naslovni dekodirnik z bralno-pisalno logiko



Slika 10: QCA implementacija sheme s slike 9

8.Signali

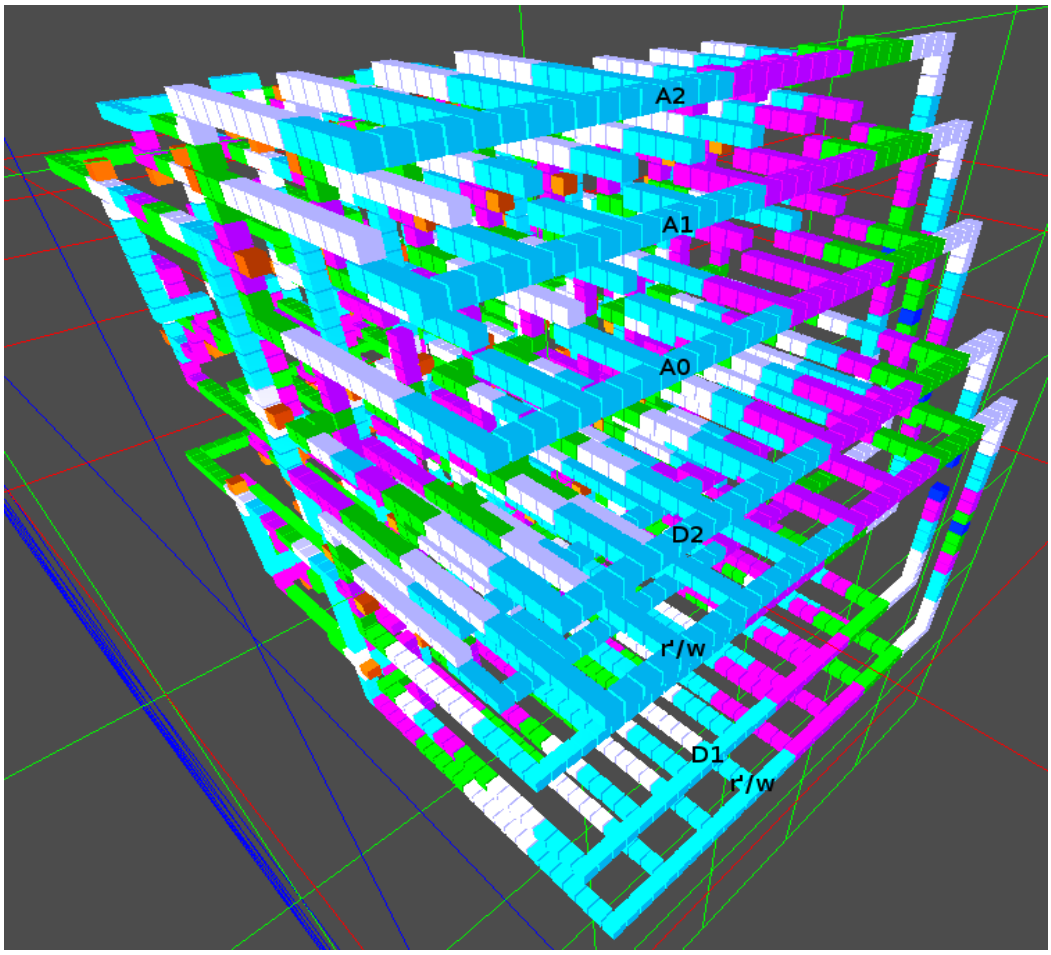
Vezje ima na vhodu naslednje signale:

- naslovne signale A0, A1, A2
- signal R/W, ki določa smer prenosa (R/W=0 določa branje, R/W=1 pa pisanje)
- dvobitno podatkovno vodilo (Din1 in Din2).

Na izhodu pa:

- dvobitno podatkovno vodilo (Dout_1 in Dout_2)

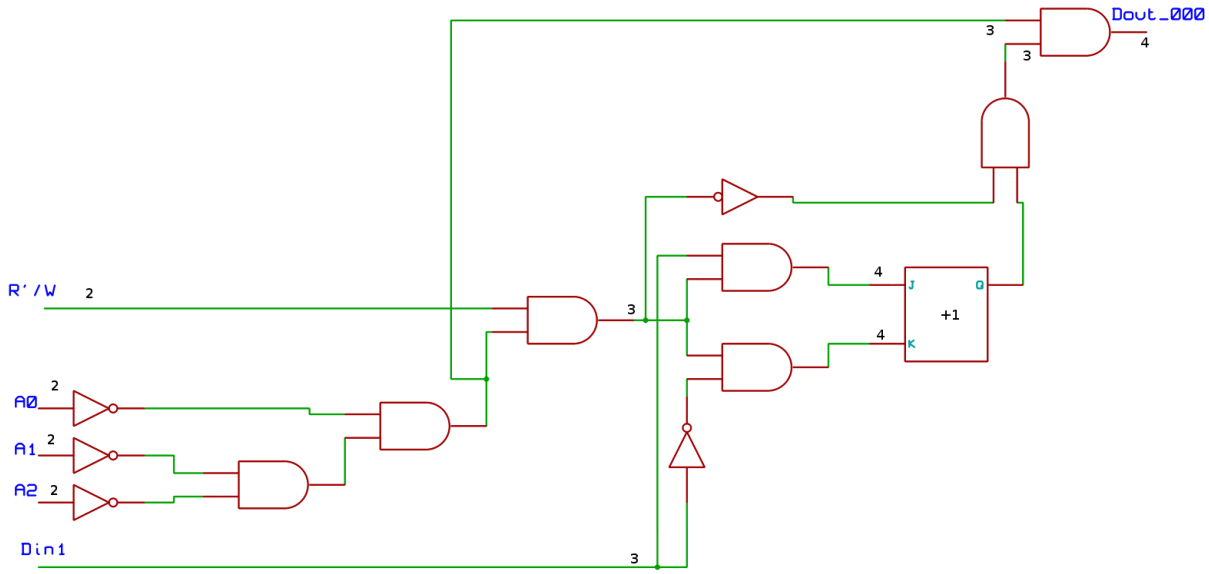
Vhodni in izhodni podatkovni signali so ločeni zaradi lažje implementacije.



Slika 11: Vhodni signali

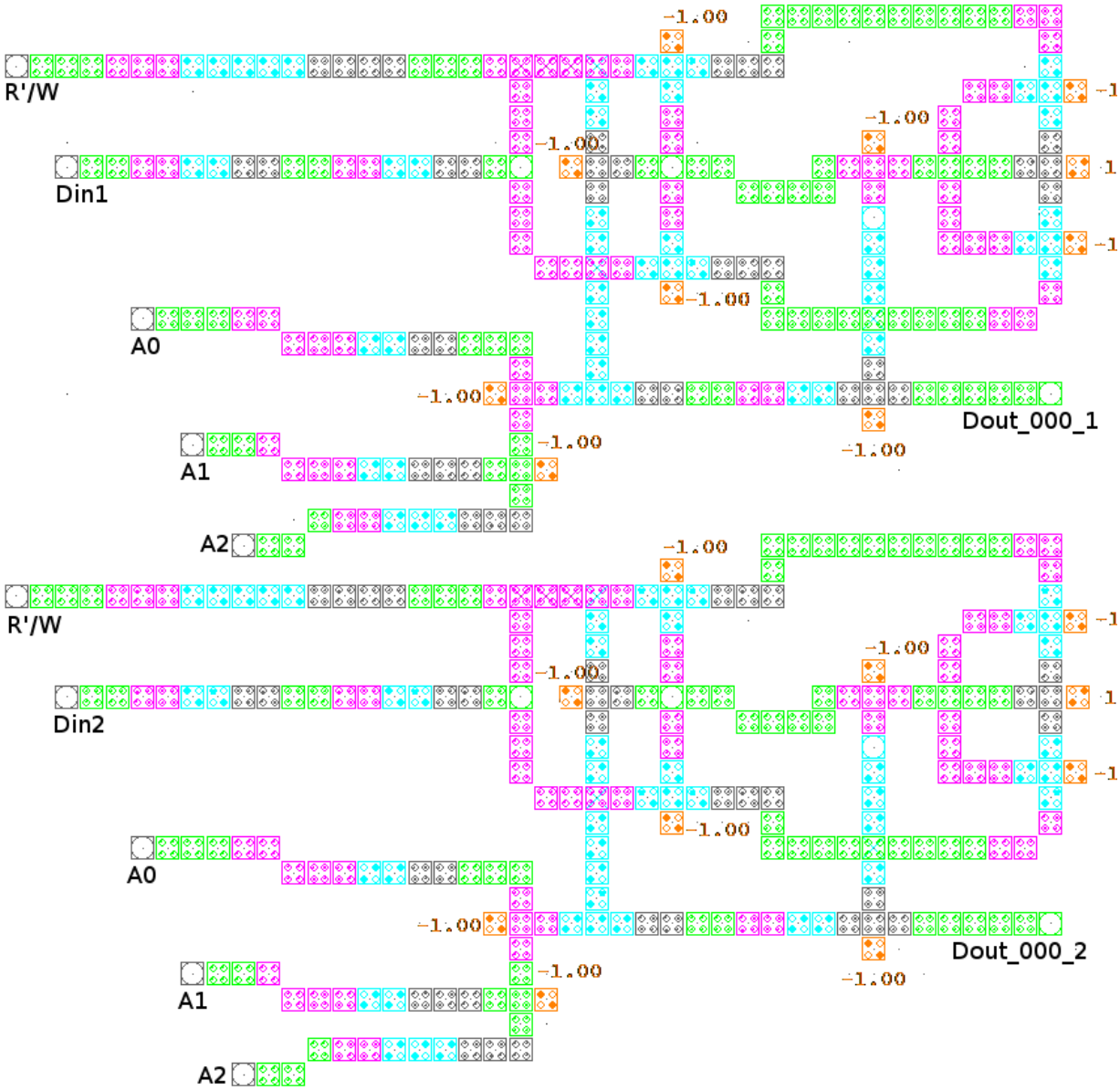
9.Realizacija pomnilne celice

Slika 12 prikazuje celico z ustrezno logiko za branje/pisanje in dekodiranje naslova. Iz slike so razvidne tudi zakasnitve. Vidimo da pisanje v celico poteka 5 urinih period. 4 urine periode signal potuje do JK celice in še ena urina perioda se porabi za spremembo stanja v celici. Branje iz celice vidimo, da poteka 4 urine periode.

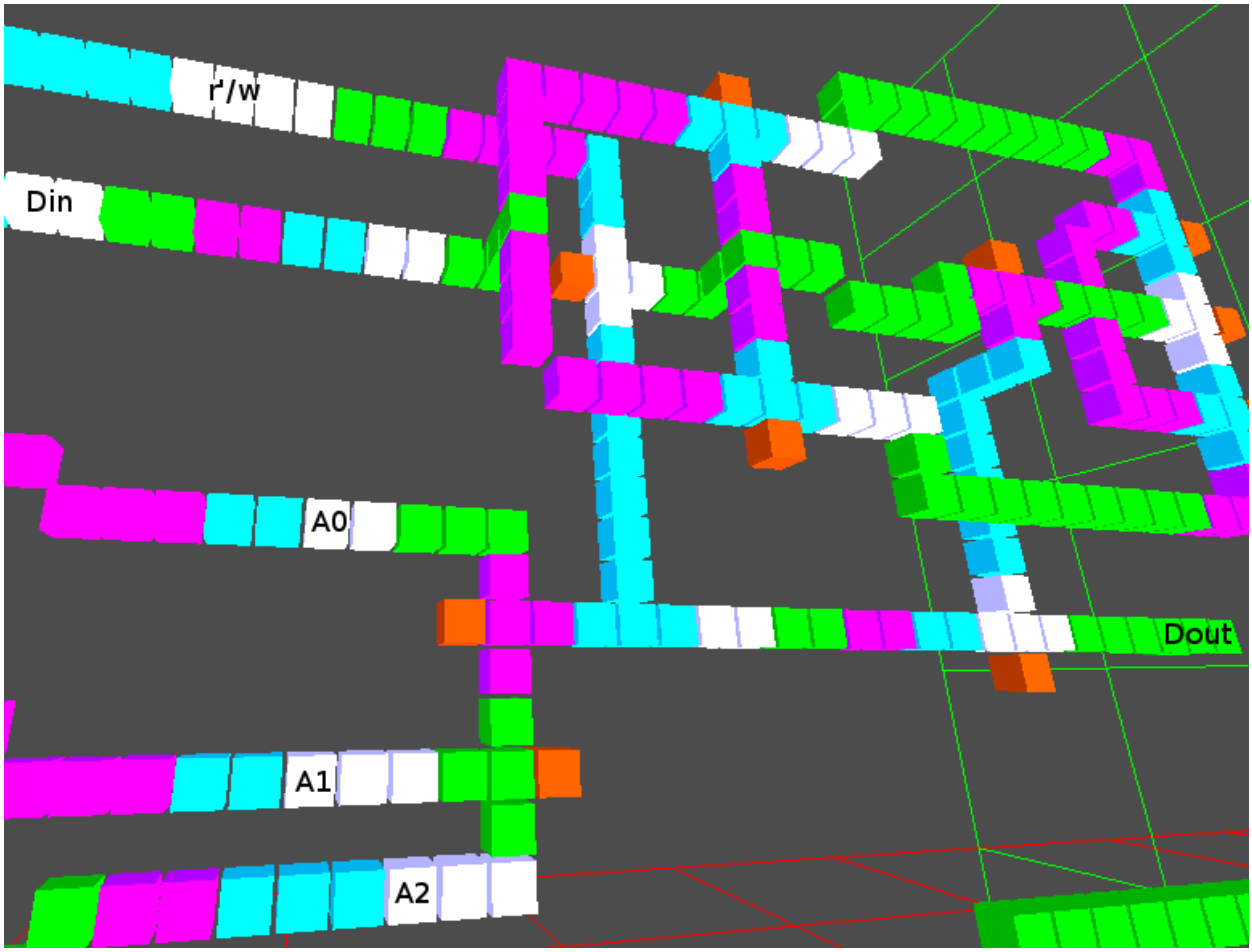


Slika 12: Celica z bralno-pisalno logiko

Slika 13 prikazuje 2 celotni logiki za dekodiranje naslova A=000. Zgornja polovica vrača podatke na izhodu Dout_1 spodnja pa na izhodu Dout_2. Tako dobimo 2 izhodna podatkovna bita. Ker je realizacija večplastna imamo na spodnjih plasteh še 7 podobnih kopij, ki imajo samo ustrezno popravljene naslovne dekodirnike. Tako dobimo skupaj 16 pomnilnih celic.



Slika 13: Implementacija slike 12 za 2-bit podatkovna vodilo

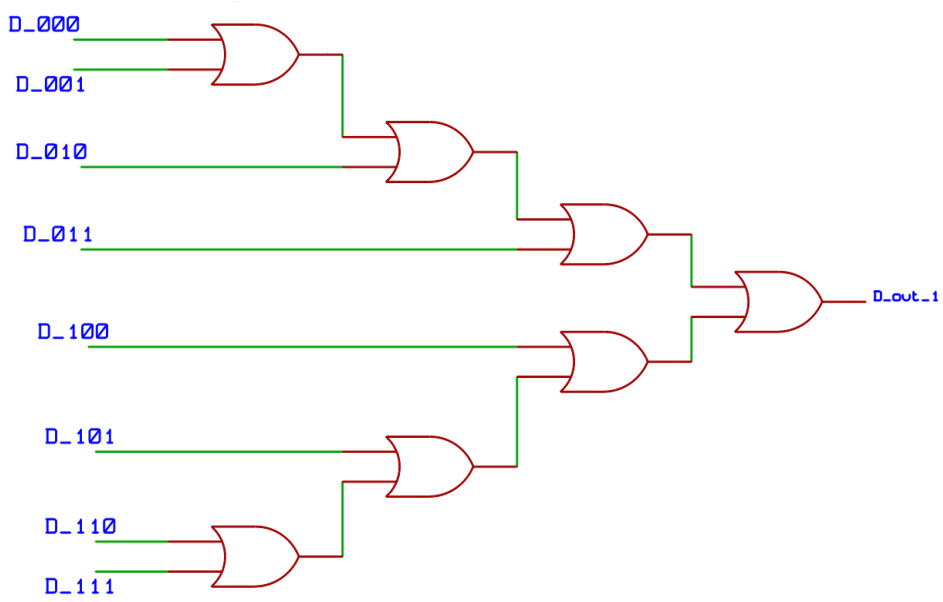


Slika 14: 3D pogled na pomnilno celico s pripadajočo logiko

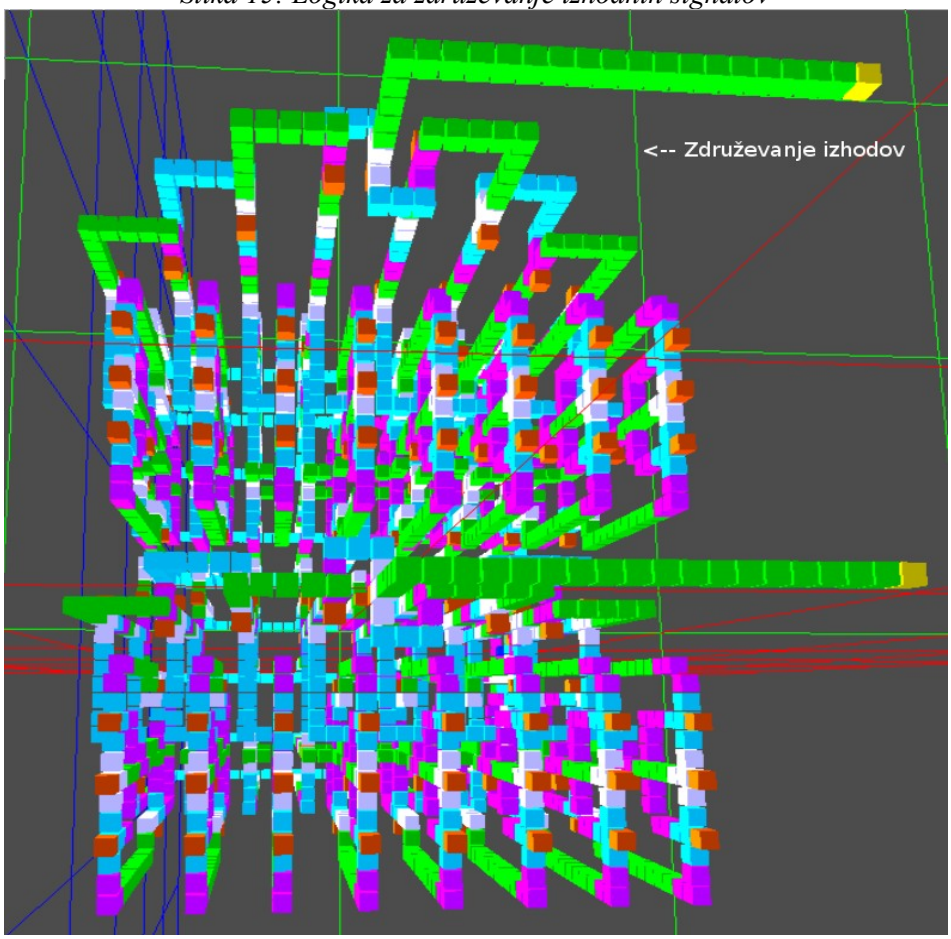
Za konec je potrebno še posamezne izhode združiti v en signal D_out_1 in D_out_2. Ker naša realizacija omogoča, da je v nekem trenutku lahko aktiven samo en izhod, ostali so 0, ne potrebujemo multiplekserja, da združimo 8 izhodov iz celic v en signal. Uporabimo lahko samo OR vrata, kar je prikazano s shemo slike 15.

Logična shema na sliki 15 prikazuje hkrati tudi kako so signali povezani med posameznimi plastmi. Vendar nam ta rešitev zakasni izhod še za 2 urini periodi kar znaša skupaj 6 urinih period, da preberemo neko informacijo iz celice. Ker v QCADesignerju opisane rešitve ne moremo vizualno prikazati dobro, nam to prikazuje slika 16.

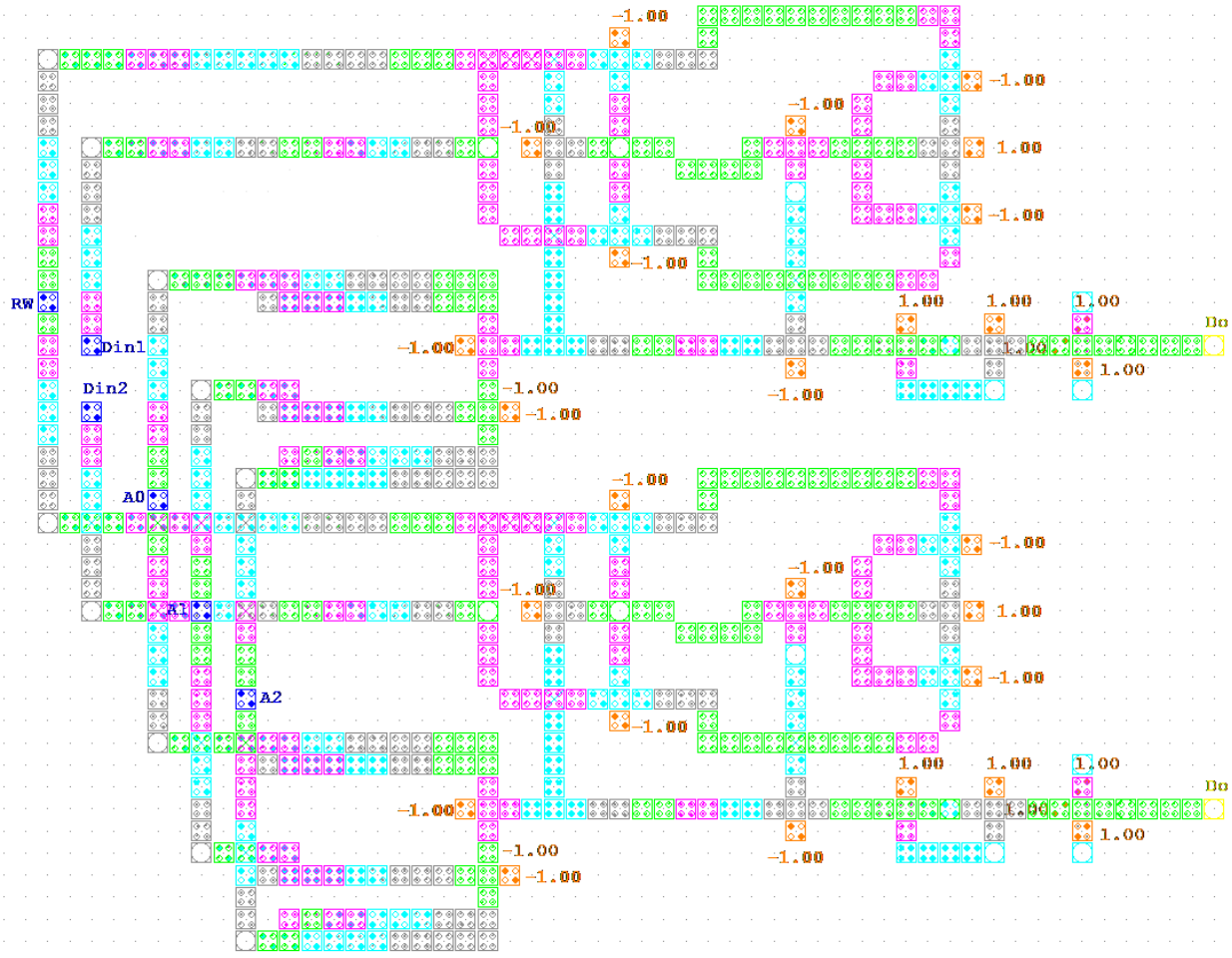
Slika 16 in 17 prikazujeta končno realizacijo dekodirnika s pripadajočimi pomnilnimi celicami, ko v QCADesigner-ju vklopimo pogled na vseh 32 plasti. Z večplastno realizacijo dosežemo boljšo preglednost nad vezjem. Ker so celice nameščene točno ena pod drugo se ne vidi obsežnosti realizacije.



Slika 15: Logika za združevanje izhodnih signalov



Slika 16:3D pogled na končno obliko dekodirnika



Slika 17: Končna oblika dekodirnika

10. Simulacija

Simulacija je bila izvedena z vnaprej določenim vhodnim vektorjem, ki je v pomnilniku prepisal vseh 8 pomnilniških besed z novimi vrednostmi nato pa dve zapisani vrednosti iz pomnilnika prebral. Med pisanjem v neko besedo in branjem iz nje morajo preteci vse 3 urine periode. Prebrana vrednost se na izhodnih linijah pojavi 6 urinih period po zahtevanem branju. Pisanje pa se zaključi 5 urinih period za izdanim zahtevkom.

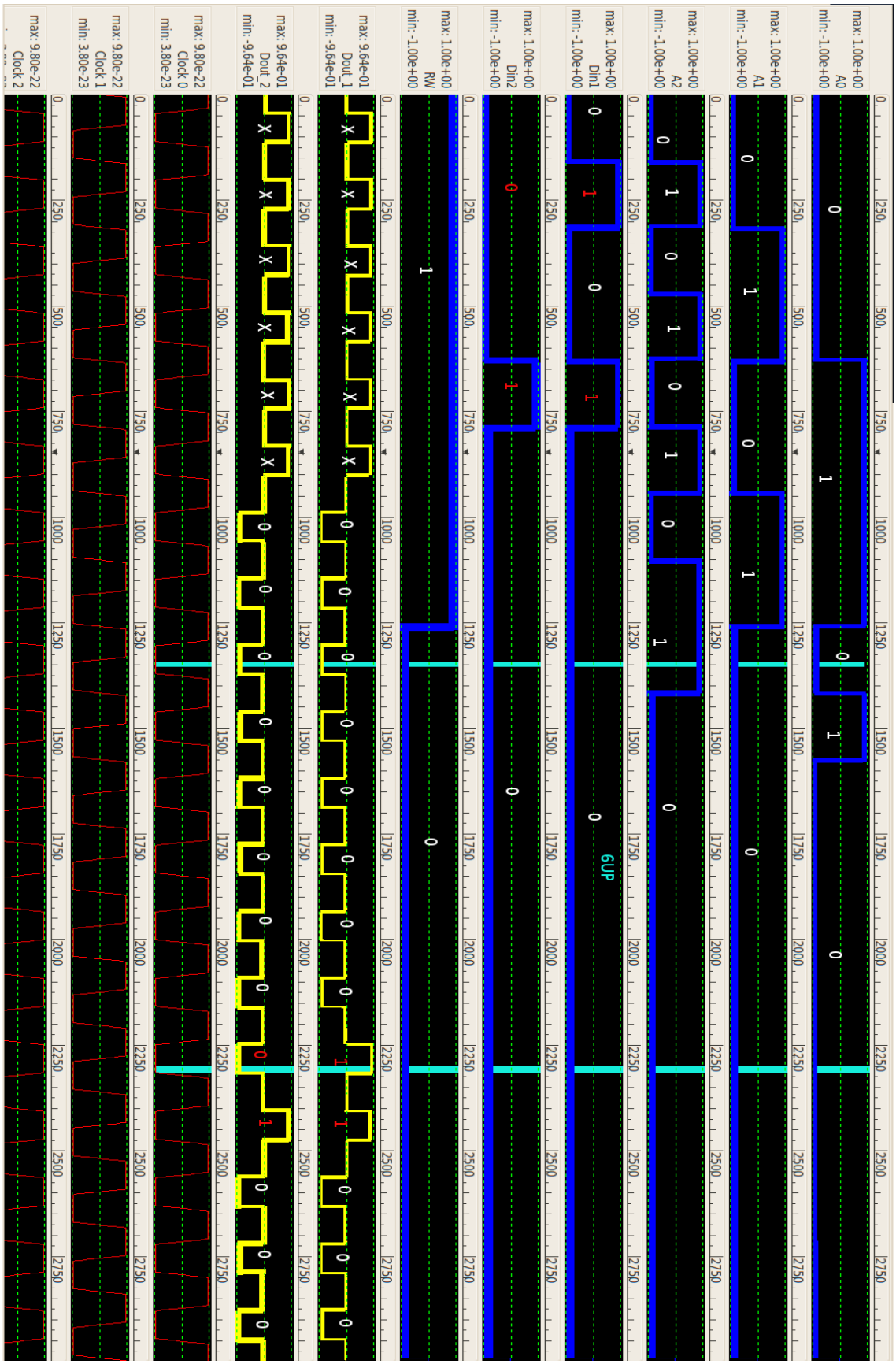
Ime signala	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
A0	0	0	0	0	1	1	1	1	0	1
A1	0	0	1	1	0	0	1	1	0	0
A2	0	1	0	1	0	1	0	1	1	0
D1	0	1	0	0	1	0	0	0	0	0
D2	0	0	0	0	1	0	0	0	0	0
R'/W	1	1	1	1	1	1	1	1	0	0

Tabela 2: Vhodni vektorji uporabljeni za testne simulacije

Tabela 2 vsebuje kombinacije vhodnih vektorjev, ki so bile uporabljene pri simulaciji delovanja. V celici 001 in 100 se vpišeta vrednosti 1 in 3 v ostale pa vrednost 0. Ko se pisanje konča pa se iz pomnilnika bere celici 001 in 100.

Na sliki 18 vidimo rezultate simulacije. Ker vezje potrebuje 6 urinih period za obdelavo vhodne informacije, je na začetku simulacije prvih 6 urinih period nedefiniranih (označeno z X). Vezje prične delovati pravilno šele po teh 6 začetnih urinih periodah. V simulaciji je prvih 8 period namenjenih pisanju vrednosti v pomnilne celice. Celice pri pisanju vedno odgovorijo z 0 na izhodu. Z 0 odgovorijo tudi vse ostale celice, ki v nekem trenutku niso naslovljene. Nato v 9 urini periodi pričnemo z branjem. Najprej beremo podatke z naslova A=100. Prvi dve rdeči številki kažeta, da smo na naslov A=100 zapisali podatek D=01 (desetiško 1). Ta podatek se nam pojavi na izhodu po 6 urinih periodah po izstavitvi ukaza za branje iz naslova A=100. Za naslov A=001 velja enako, kjer je vpisan podatek D=11 in potem tudi za naslov A=000, v katerega je vpisano D=00 (zadnje tri ničle na izhodu).

Simulacija je bila izračunana s pomočjo „Coherence vector“.



Slika 18: Nivoji signalov pri testni simulaciji

11. Zaključek

Pri realizaciji danega vezja smo se srečali z veliko težavami. Izkazalo se je, da nekega problema ni težko logično realizirati v QCA ampak se težave pojavijo pri povezovanju že v naprej narejenih logičnih enot. Samo celico, ki je sposobna pomniti in sprejemati signala RW in Din ni bilo težko realizirati. Vendar ko je bilo potrebno na to pomnilno celico pripeljati ustrezne signale, so se pa pojavili problemi. Ena od najbolj problematičnih elementov je bila dolga linija, kjer se vhodna vrednost ne pojavi tudi na izhodu. Zato je potrebno linijo ustrezno razdeliti na „clock zone“, da se nam signal ne oslabi. Slabost tega je, da se zakasnjuje s tem tudi delovanje vezja, kar pa v računalništvu ni najbolj zaželena stvar.

Naše vezje je sposobno zapisati nek podatek v 5 urinih periodah in ga prebrati v 6 urinih periodah. Zakasnitve bi bilo možno izboljšati, a za več kot 1 urino periodo zelo težko, saj sam osnovni načrt tega ne dovoljuje zlahka.

12. Viri

- prof. Miha Mraz: predloge za predavanja pri predmetu ONT
- mag. Primož Pečar: predloge za vaje pri predmetu ONT
- Electronic tutorials: [http://www.electronics-tutorials.ws/combination/comb_5.html]
- Wikipedia: [http://en.wikipedia.org/wiki/Address_decoder]
- The University of British Columbia: [http://www.mina.ubc.ca/iwqca]