Univerza v Ljubljani

Optične in nanotehnologije

1.SEMINARSKA NALOGA

4-bitni hitri pomikalnik

Matej Pangerc,63060290,
Grega Škaper,63050112,
Leon Golob,63060052,
Gašper Cvenkel, 63050019

Ljubljana, 26. 11. 2010

Kazalo

Uvod 1

Implementacija 4-bitnega hitrega pomikalnika 2

Funkcije hitrega pomikalnika 2

Osnovna logična shema vezja 4

Realizacija s QCADesigner-jem 5

Osnovni gradniki vezja 8

Analiza zakasnitve vezja 9

Problemi in posebnosti pri realizaciji 10

Rezultati testiranj 11

Zaključek 13

Literatura 14

Uvod

Pri predmetu Optične in nanotehnologije smo z orodjem QCADesigner, ki je odprtokodno simulacijsko orodje za delo s kvantnimi celičnimi avtomati, implementirali 4 bitni hitri pomikalnik, ki zna pomikati v levo in desno za n mest (n<=4). Ločeno smo realizirali krožnega, logičnega in aritmetičnega. Problema smo se najprej lotili z realizacijo D-pomnilnih celic, vendar je to rešitev le za pomikalnik, kar pomeni da bi rabili minimalno 4 urine periode. Zaradi hitrega pomikalnika potrebujemo rezultat že v naslednji periodi, zato smo nalogo realizirali z multiplekserji.

Implementacija 4-bitnega hitrega pomikalnika

Funkcije 4-bitnega hitrega pomikalnika

Pri implementaciji hitrega pomikalnika smo najprej napisali pravilnostne tabele, iz katerih smo razbrali, kakšni morajo biti izhodi za posamezne vhode.

Pravilnostna tabela za krožni pomikalnik za vhodni vektor A3 A2 A1 A0:

S2	S1	S0	D3	D2	D1	D0
0	0	0	A0	A3	A2	A1
0	0	1	A1	A0	A3	A2
0	1	0	A2	A1	A0	A3
0	1	1	A3	A2	A1	A0
1	0	0	A2	A1	A0	A3
1	0	1	A1	A0	A3	A2
1	1	0	A0	A3	A2	A1
1	1	1	A3	A2	A1	A0

Tabela 1: Pravilnostna tabela krožnega hitrega pomikalnik

Z **S1** in **S0** določamo za koliko naj se vektor pomakne:

S1	S0	POMIK
0	0	1
0	1	2
1	0	3
1	1	4

Tabela 2: Določanje premika v levo ali desno

Z **S2** pa določamo ali naj pomikalnik pomika v desno (S2=0) ali v levo (S2=1).

Pravilnostna tabela za logični pomikalnik za vhodni vektor A3 A2 A1 A0:

S2	S1	S0	D3	D2	D1	D0
0	0	0	0	A3	A2	A1
0	0	1	0	0	A3	A2
0	1	0	0	0	0	A3
0	1	1	0	0	0	0
1	0	0	A2	A1	A0	0
1	0	1	A1	A0	0	0
1	1	0	A0	0	0	0
1	1	1	0	0	0	0

Tabela 3: Pravilnostna tabela za logični hitri pomikalnik

Vhodi **S2**, **S1**, **S0** imajo enako funkcijo, kot opisano pri krožnem pomikalniku.

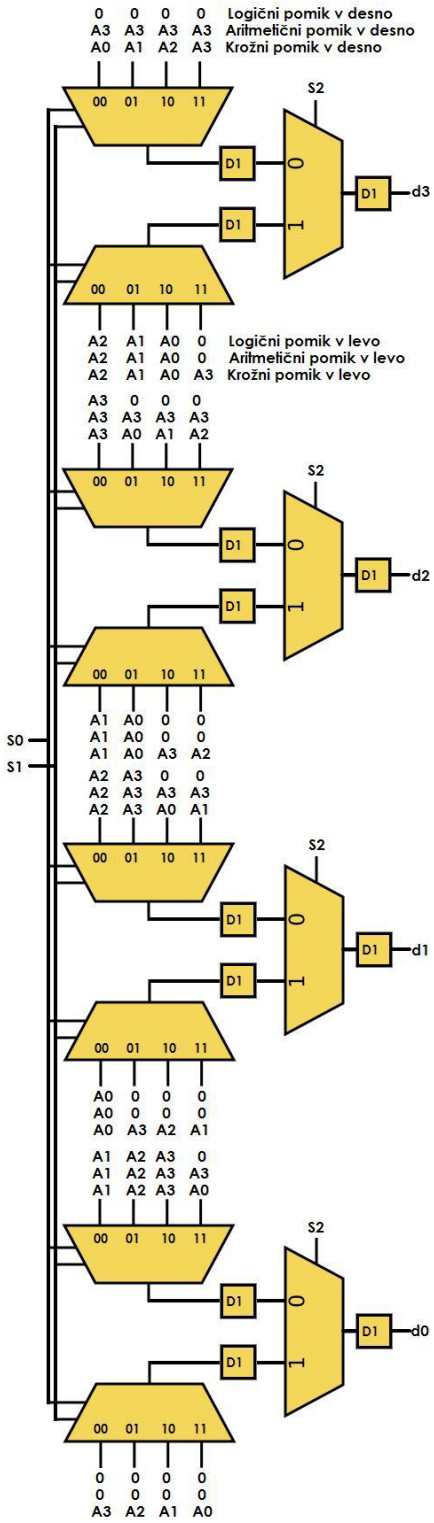
Pravilnostna tabela za aritmetični pomikalnik za vhodni vektor A3 A2 A1 A0:

S2	S1	S0	D3	D2	D1	D0
0	0	0	A3	A3	A2	A1
0	0	1	A3	A3	A3	A2
0	1	0	A3	A3	A3	A3
0	1	1	A3	A3	A3	A3
1	0	0	A2	A1	A0	0
1	0	1	A1	A0	0	0
1	1	0	A0	0	0	0
1	1	1	0	0	0	0

Tabela 4: Pravilnostna tabela za aritmetični hitri pomikalnik

Vhodi **S2**, **S1**, **S0** imajo enako funkcijo, kot opisano pri krožnem pomikalniku.

Osnovna logična shema vezja



Slika 1: Logična shema vezja

Slika 1 prikazuje osnovno logično shemo vezja, po kateri smo se zgledovali pri izdelovanju našega pomikalnika. Odvisno od tipa vezja in od željenega izhoda pripeljemo na vhode mux-ov različne vhode (glej tabele). Spodnji mux je aktiven, ko pomikamo v levo, zgornji pa ko pomikamo v desno. Shema je narejena za samo en izhod, mi pa smo zadevo razširili, tako da smo dobili vse 4 izhode naenkrat.

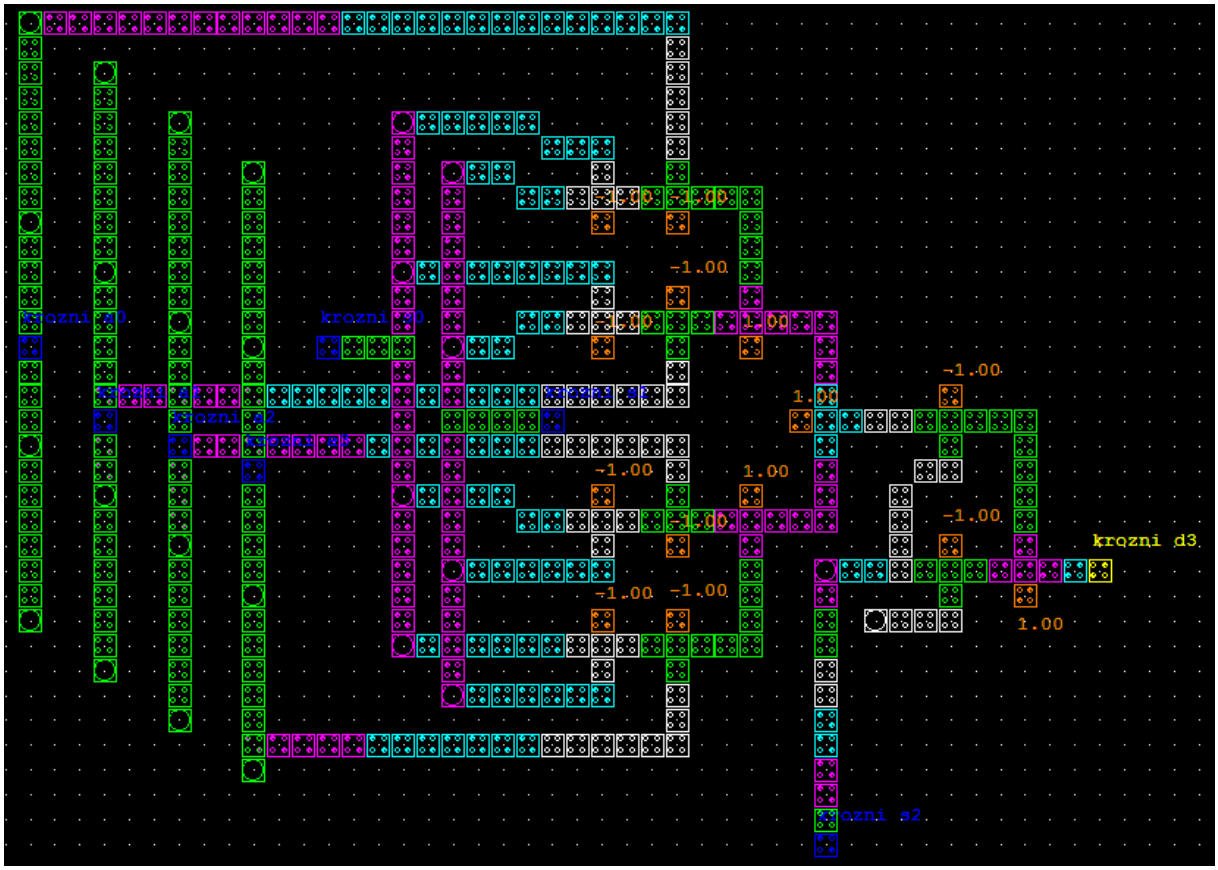
Realizacija z QCADesigner-jem

Za realizacijo smo uporabili 6554 celic, katere smo razdelili v 17 plasteh. Najvišja plast so vsi ločeni vhodi tako za aritmetični, logični in krožni pomikalni. Sledijo pa še štiri krat po štiri plasti za posamezni izhod, saj imamo 4-bitni pomikalnik.

Za en izhod rabimo 4 plasti. Dve plasti sta ločevalni, na ostalih dveh pa je na eni realizacija enega 4:1 MUX ki pride povezan na zadnjo plast, kjer je še drugi 4:1 MUX in 2:1 MUX. To je realizacija za en izhod **D** našega pomikalnika. Časovni zamik pride za 2 urini periodi in pol, kar pomeni da za pravilen rezultat gledamo clock2.

Zaradi velike količine plasti in ker je dejansko realizacija po plasteh enak za vse štiri izhode, bomo v slikah prikazali le zgornjo plast in povezave.

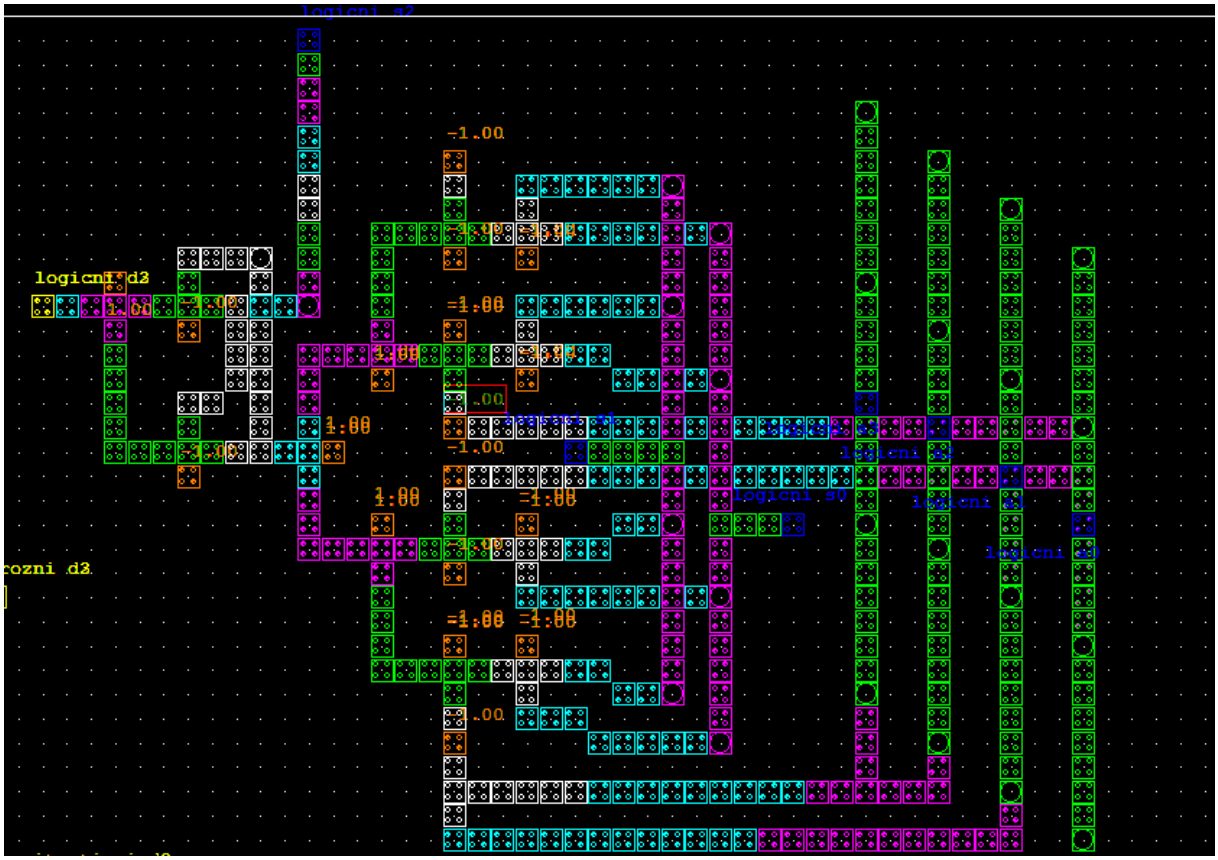
Krožni pomikalnik:



Slika 2: Zgornji nivo enega izhoda za krožni pomikalnik

Pri logičnem hitrem pomikalniku se realizacija razlikuje od krožne po tem da na posameznih nivojih na vhode peljemo ničle, kjer je le to glede na pravilnostne tabele potrebno.

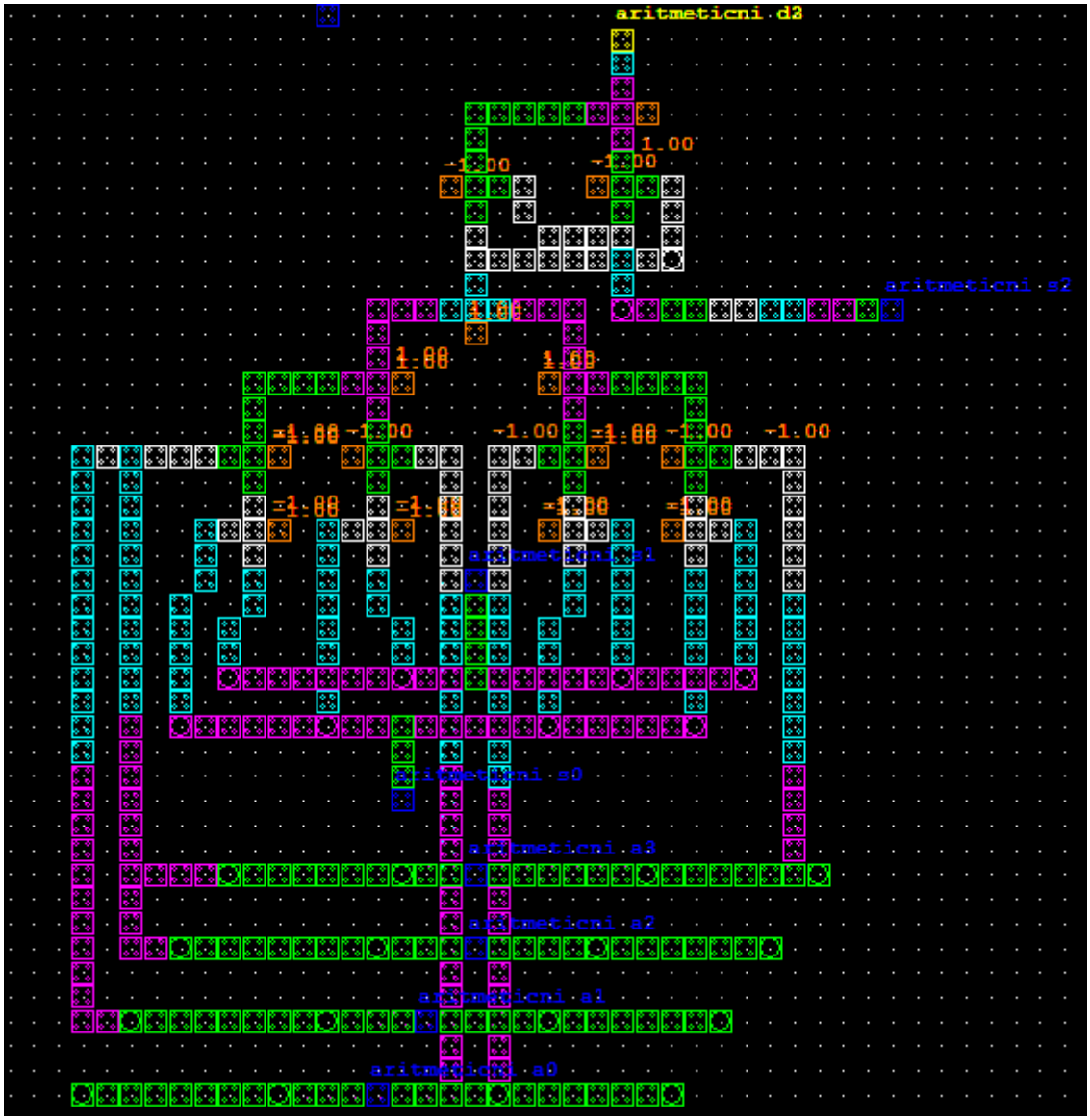
Logični pomikalnik:



Slika 3: Zgornji nivo enega izhoda za logični pomikalnik

Aritmetični pomikalnik pa v nasprotju z logičnim, za premik v desno potrebuje premikati oziroma dodajati takšno vrednost, kot je vrednost vodilnega bita.

Aritmetični pomikalnik:



Slika 4: Zgornji nivo enega izhoda za aritmetični pomikalnik

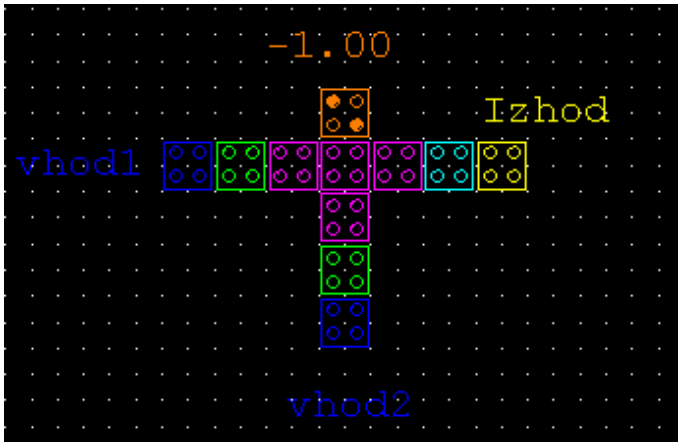
Osnovni gradniki vezja

Pri realizaciji vezja smo uporabljali:

- ***Negator***
  Negator je osnovni gradnik, unarni operator, kateri nam negira vhodno vrednost.

Slika 5: Negator

- ***Majoritetna vrata (and vrata)***
  Primer na sliki nam prikazuje **AND** vrata. Sestavljena so iz dveh vhodov, konstante in izhoda. Pri tem moramo paziti na pravilno postavitev ure. Vhodi morajo biti enako zakasnjeni. Osrednjih 5 celic mora imeti en urin zamik več kot oba vhoda in enega manj kot izhod iz majoritetnih vrat.
  Razlika med AND in OR vrati je samo v konstanti, katera je pri OR enaka 1.



Slika 6: Majoritetna vrata

Multiplekser 4:1 je sestavljen iz osmih AND majoritetnih vrat, ter treh OR vrat, multiplexer 2:1 pa iz dveh AND vrat in enih OR vrat. Negatorji so bili uporabljeni le za pravilno pripeljane vhode glede na logično fukcijo.

Analiza zakasnitve vezja

Zaradi predolgih povezav do majoritetnih vrat(AND, OR) smo povezave do njih različno zakasnili. Uporabili smo štiri urine signale med katerimi je četrtinski fazni zamik. Najprej smo začeli s prvim urinimi signalom(clock 0), nadaljevali z drugim(clock 1), tretjim(clock 2) in končali z četrtim(clock 3).

- zakasnitev 4:1 mux vezja

Povezavi S0 in S1 smo zakasnili s četrtim urinim signalom(clock 3), ko smo jih povezali z AND vrati. Povezavo iz AND vrat povežemo na naslednja AND vrata(clock 0) v katere pripeljemo povezavo iz vhodov(A0,A1,A2,A3), ki je zakasnjena za clock3. Povezave iz AND vrat sledijo OR vratom(clock 1). OR vrata povežemo na naslednji OR (clock 2). Povezava zakasnjena za clock 3 pripeljemo na vhode AND vrat(clock 2) 2:1 muxa. Tako vidimo da je naš celoten 4:1 mux z vsemi vhodnimi povezavami v celoti zakasnjen za 2 urini periodi.

- zakasnitev 2:1 mux vezja

Povezava S2 vhoda in 4:1 mux pripeljemo na vhod(clock 3) AND vrat z clock 0, vse skupaj povežemo v OR vrata z clock 1. Izhod iz 2:1mux zakasnimo še za clock 2. Tako dobimo podatek na izhodu zakasnjen za 2.5 urini periodi urinega signala clock 2.

Problemi in posebnosti pri realizaciji

Izhod vezja je glede na vhod zamaknjen za 2.5 urine periode. S tem se stanje celic na linijah stabilizira, poveča se zanesljivost in doseže pravilnost delovanja vezja.Prva težava na katero smo naleteli je bila postavitev vhodov oziroma njihovo povezovanje. Zaradi tega smo celotno nalogo naredili na plasteh, ker so povezave tako manjše. S tem smo se izognili predolgim linijam in posledično tudi daljšim urinim periodam. Tudi sami vhosi so realizirani na svojem nivoju.

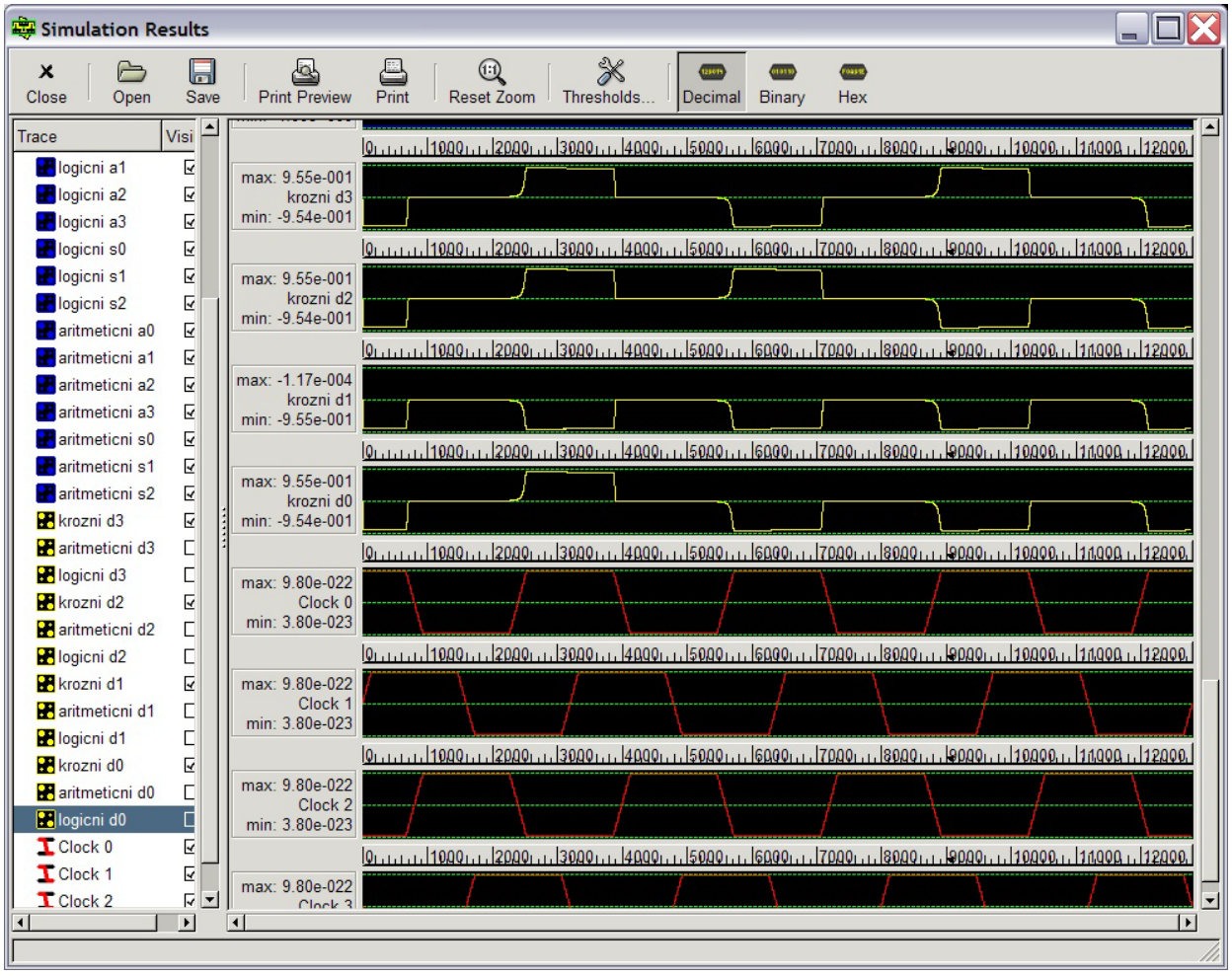
Eden od problemov je bil tudi kako celotne multiplekserje postaviti na nivoje. Dileme so bile ali naredimo vse tri oblike hitrega pomikalnika nivojsko, kar bi nam prineslo okoli 51 nivojev. Druga opcija je bila da bi dejansko naredili tri logične qca file-e. Na koncu smo vse naredili v enem file-u, ter vse 3 oblike na nivojih. S tem pa smo dobili veliko vhodov in tudi izhodov; posamezni vhodi in izhodi za vsake posamezne oblike hitrega pomikalnika.

Težave so bile tudi pri samem debugiranju, saj je napake zelo težko odpravljati. Zlasti pri večnivojskih strukturah je animirani simulaciji precej težko slediti.

Rezultati testiranj

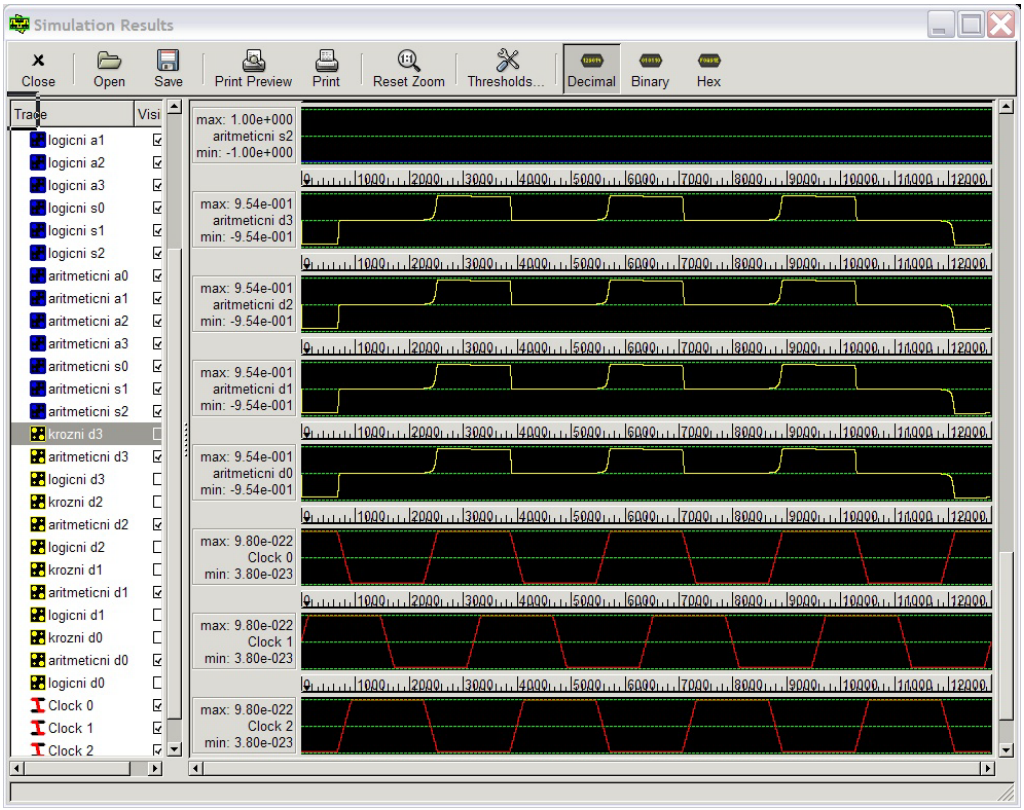
Na sliki vidimo rezultat simulacije, ki smo jo pognali v QCA Designerju. Pravilnost delovanja vezja smo preverjali z vhodnimi vektorji, ker je bilo tako najlažje odkriti napake. Pravilen rezultat je na izhodu zamaknjen za 2.5 urine periode.
Na slikah je pri vseh treh oblikah hitrega pomikalnika dan primer testiranja z vhodnim vektorjem 1 0 0 0. Premik je bil v desno smer za n=4; S0=1,S1=1, S2=0.

Rezultati testiranj za krožno obliko hitrega pomikalnika nam da rezultat: 1 0 0 0



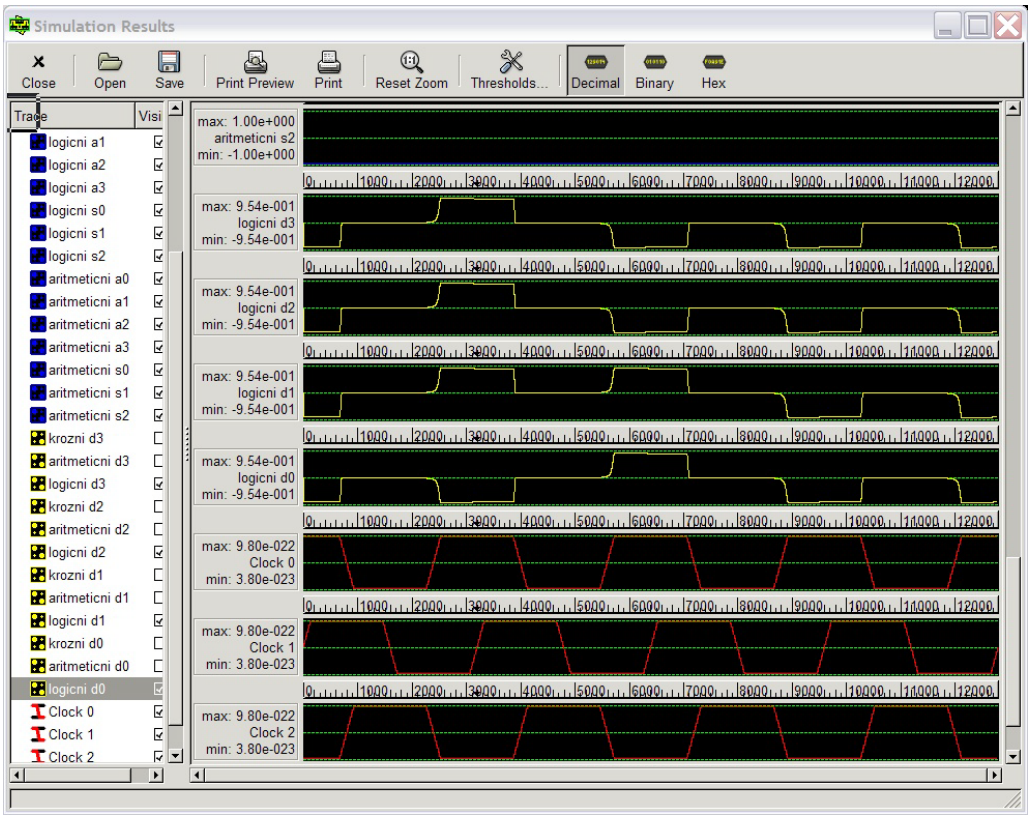
Slika 7: Rezultat simulacij za krožni pomikalnik

Rezultati testiranj za aritmetično obliko hitrega pomikalnika nam da rezultat: 1 1 1 1



Slika 8: Rezultat simulacija za aritmetični pomikalnik

Rezultati testiranj za logično obliko hitrega pomikalnika nam da rezultat: 0 0 0 0



Slika 9: Rezultat simulacij za logični pomikalnik

Zaključek

S QCA Designer-jem smo se srečali prvič, zato smo imeli na začetku nemalo težav. Ne toliko zaradi logike, ampak zaradi povsem novega nivoja fizičnih lastnosti in medsebojnega vpliva posameznih celic. Zaradi opisanih problemov, nas je to pripeljalo do takega načina realizacije.

Razmišljali smo da bi lahko poleg te realizacije naredili še dve različni verziji. Da bi imeli vse vhode in izhode povezane, torej bi dejansko minimizirali naš vhodno/izhodni 15 sistem, vendar bi pridobili na urinih periodah. Tretji način bi pa uporabili samo eno strukturo multiplekserjev (8 krat 4:1 Mux in 4 krat 2:1 Mux, namesto 24 krat 4:1 Mux in 12 krat 2:1 Mux), kjer bi pred vsakim vhodom multiplekserjev izbirali eno od treh možnih oblik hitrega pomikalnika. Pri tem bi pred vsakim 4:1 Mux porabili štiri 3:1 Mux . Kar pa nas privede do bolj kompleksnega vezja ni večjega števila gradnikov.

Tako vidimo da pri vseh treh realizacijah nekaj pridobimo in nekaj izgubimo. Mi smo se odločili za večje število vhodov/izhodov, seveda pa je vezje glede urinih period krajše in manj kompleksno.

Literatura

Power point prezentacija realizacije hitrega pomikalnika z multiplekserji:

http://users.ece.gatech.edu/~leehs/ECE2030/slides/**Lec13-shifter.ppt**

Primer že narejene Shift enote:

http://web.cecs.pdx.edu/~mperkows/CLASS_FUTURE/QDCA/calgary-arithmetic-layout-ettc_qca.pdf

Wikipedij:

http://en.wikipedia.org/wiki/Multiplexer