Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

1.seminarska naloga

4-Bitni pomikalnik (BARREL SHIFTER)

Realizacija s kvantnimi celularnimi avtomati
[6.skupina]

OPTIČNE IN NANOTEHNOLOGIJE

Borut Ajdič, Marko Gavrilovič,

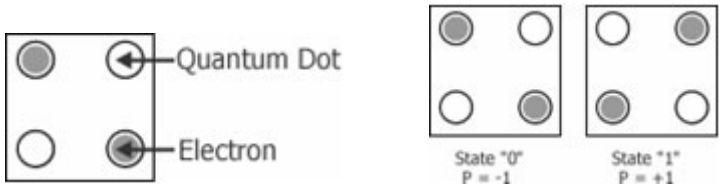
Tine Ileršič, Simeon Puntar

Ljubljana, december 2010

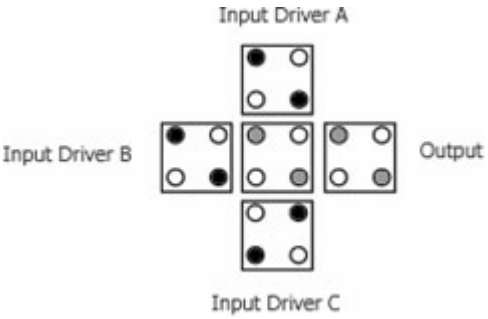
Kazalo

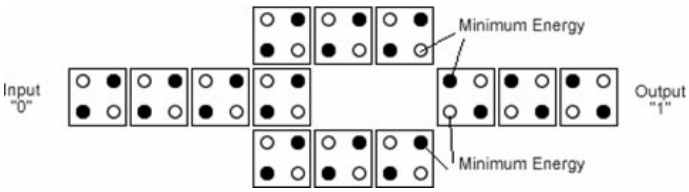
Uvod

QCA (kvantni celični avtomati) so eden od modelov kvantnega računanja. Njihova osnovna enota je celica iz 4 kvantnih pik in 2 elektronov. Zaradi elektrostatskega odboja lahko elektrona zasedeta dve različni stanji. Eno od njih predstavlja logično 0, drugo pa 1.



S QCA lahko realiziramo funkcijsko poln sistem logičnih funkcij. Njihova osnovna vrata so tri vhodna majoritetna vrata in not vrata. Z majoritetnimi vrati lahko realiziramo funkciji and (enega od vhodov postavimo na -1) in or (enega od vhodov postavimo na 1).





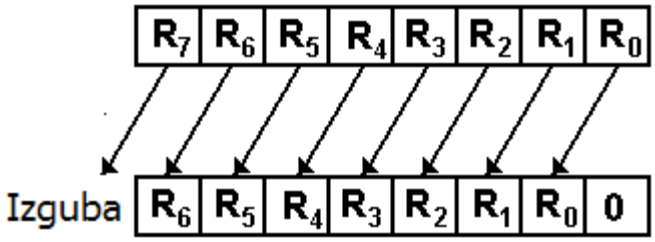
Simuliranje delovanja QCA nam omogoča odprtokodno orodje za delo s kvantnimi celičnimi avtomati. V tem orodju bomo tudi poizkušali realizirati 2 in 4-bitne pomikalnike oz. barrel shifterje.

Predstavitev problema

V QCA designerju smo morali realizirati 2 in 4-bitni Barrel shifter. Omogočati mora pomike v levo in desno za 0-1 mesto pri 2-bitnem in 0-3 mesta pri 4-bitnem pomikalniku. Pomiki so lahko logični, aritmetični in krožni.

Logični pomikalnik

Logični pomikalnik deluje tako, da premika bitne vrednosti v željeno smer, na izpraznjena mesta pa dodaja logično 0. Pomiki so lahko poljubno celo število, toda pomik za več kot N mest, pri čemer je N dolžina registra, povzroči, da se nastavijo same logične 0.

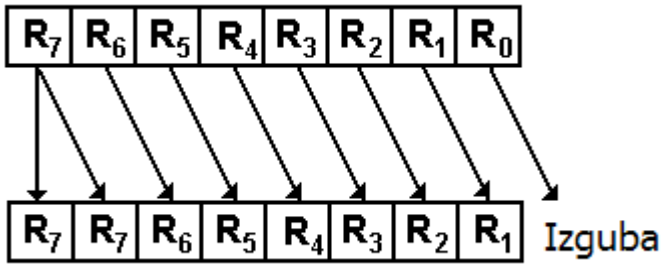


Primer pomikanja v 8-bitnem registru: 1001 0110 0010 1100

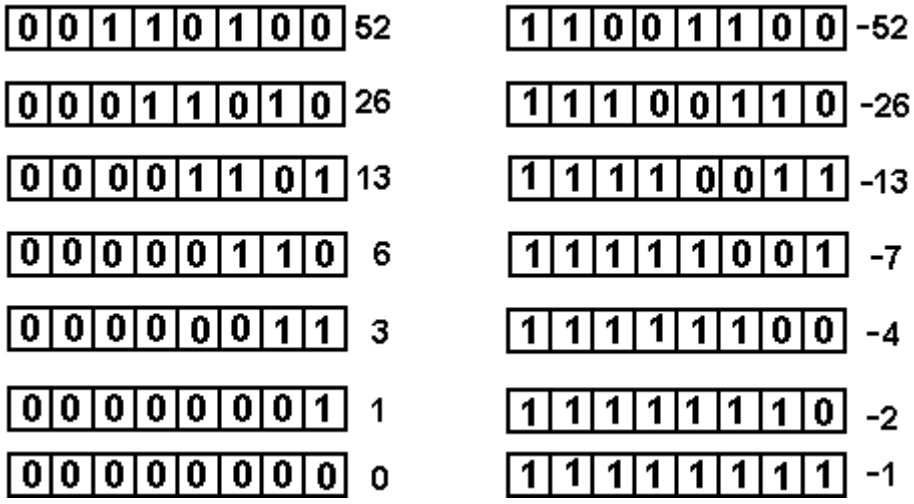
Običajno pravilo za N-bitni register je, da je število pomikov med 0 in vključno (N-1).

Aritmetični pomikalnik

Aritmetični pomikalnik deluje enako kot logični, le da pri tem ohranja zadnji bit, ki označuje predznak.

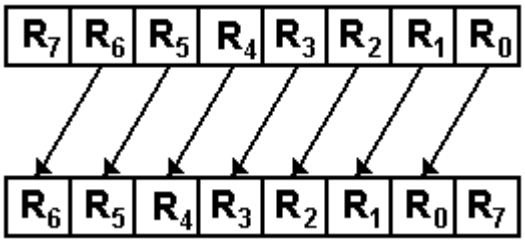


Namen aritmetičnega pomikalnika je pri pomikanju v desno ustvariti ekvivalenco deljenju s številom 2 v dvojiškem komplementu in pri pomikanju levo množenje s številom 2. Za primer vzemimo 8-bitni register in število 52 (0011 01002).



Krožni pomikalnik

Krožni pomikalnik je identičen logičnemu pomikalniku, le da se pri tem izpadli biti vrnejo na drugo stran.

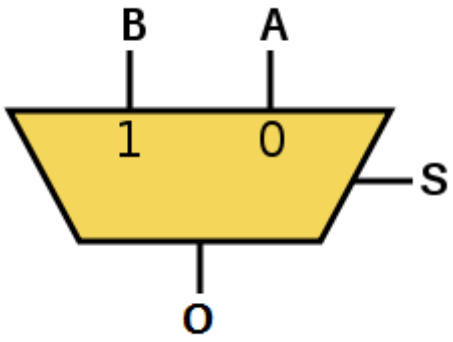


Krožni pomikalniki se veliko uporabljajo pri pisanju gonilnikov naprav. Gonilniki označujejo bite kot logične zastavice. Npr. da bita 3 in 2 v 8-bitnem registru označujeta izbrano napravo. Pomaknemo za 2 mesti v desno, da poravnamo bite in izvedemo logičen AND. Opazimo tudi, da krožni pomik ne izgublja informacije.

R =	0010 0101	
Pomik v desno za 2	0100 1001	
AND 0000 0011	0000 0001	Izbrana naprava je 01.

Realizacija

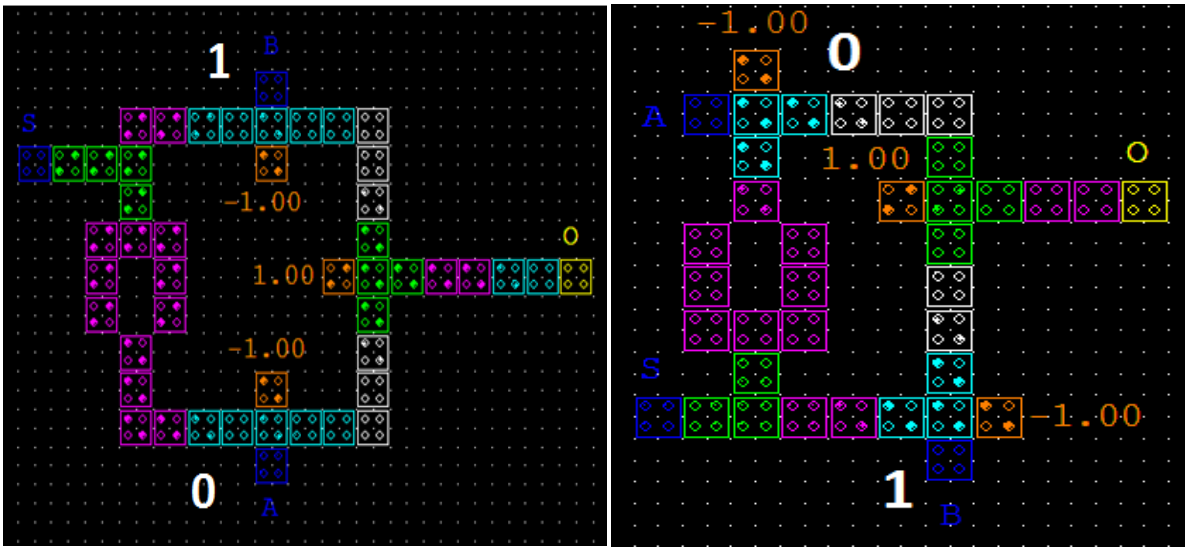
Pri realizaciji smo se odločali med več različnimi rešitvami, na koncu smo se odločili za povezavo več 2/1 multiplekserjev.



Multiplekser 2/1

2/1 multiplekser ima vhoda A in B, kontrolni vhod S in izhod O. Pri njem s kontrolnim vhodom S izbiramo med vhodoma A in B. Če je S=1, se na izhodu pojavi vrednost B, če je S=0, se na izhodu pojavi vrednost A.

Pri izvedbi s QCA smo preizkusili več oblik multiplekserja, vendar je večina med njimi delovala nepredvidljivo. Manj težav smo sicer imeli pri 2-bitni realizaciji, z naraščanjem števila multiplekserjev pri 4-bitni realizaciji pa se je zmanjšala stabilnost. Uporabljali smo dve različni izvedbi multiplekserjev, ki sta se izkazali kot najbolj stabilni in zanesljivi.



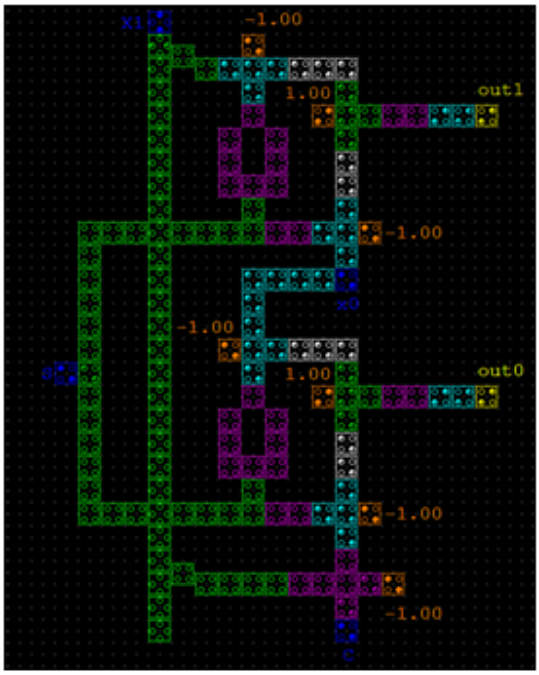
V prvi urini periodi se izvede negacija S ter dvakrat operacija AND. Na začetku druge urine periode se izhoda iz obeh AND vrat pripeljeta na OR vrata, izhod teh vrat pa je O.

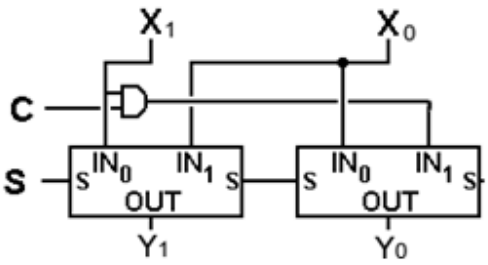
2-bitni pomikalniki

Po odločitvi za osnovno strukturo smo najprej realizirali 2-bitne logične, aritmetične in krožne pomikalnike v levo in desno za 0-1 mesto. Vsi uporabljajo dva 2/1 muxa z enim krmilnim signalom S.

Krožni/logični pomikalnik v levo

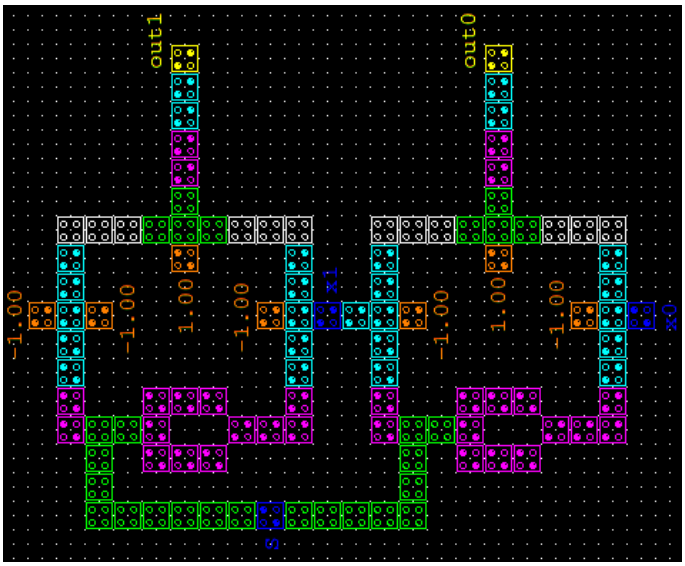
Najprej smo realizirali logični pomik, kasneje smo dodali še možnost krožnega pomika oziroma izbire med logičnim pomikom in krožnim pomikom. Med njima izbiramo z bitom C, če je bit C=1, imamo krožni pomikalnik, če C=0, pa pomik v levo.





Logični pomikalnik v desno

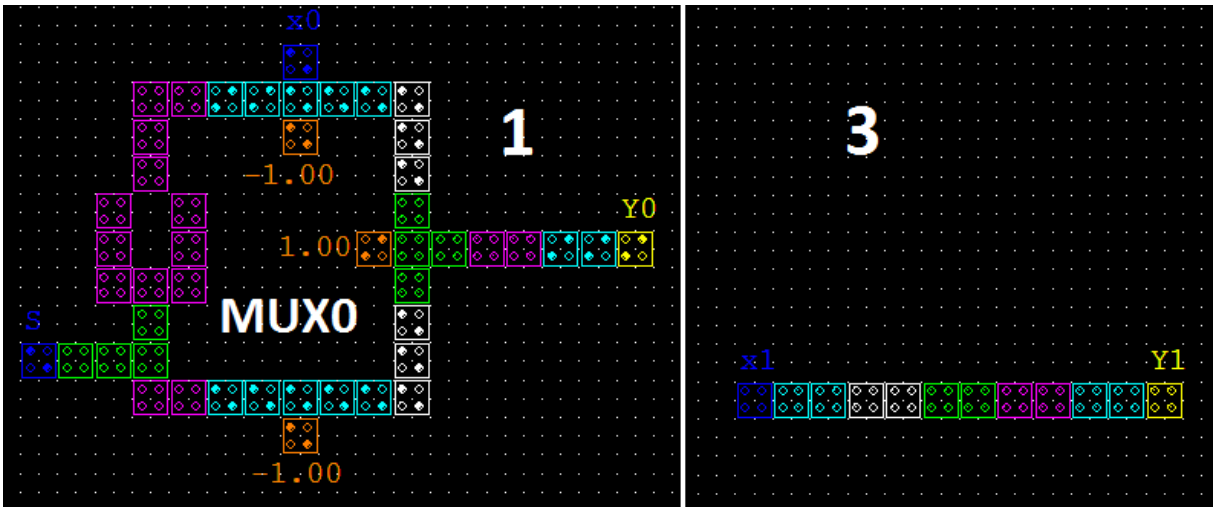
Logični pomikalnik v desno je skorajda zrcalna slika logičnega pomikalnika v levo. Prvi bit se pri vključenem pomiku izgubi oziroma gre na njegovo mesto 0, sam pa se premakne v desno na mesto zadnjega bita.



Aritmetični pomikalnik v desno

Aritmetični pomikalnik v desno deluje tako, da je ne glede na kontrolni vhod S stanje izhoda Y1 enako vhodu x1. Zato smo uporabili samo en multiplekser, katerega naloga je, da ob aktivnem signalu S da na izhod x1, ob neaktivnem pa na izhod Y0 prepiše x0. Na izhod Y1 smo samo povezali izhod x1 in ga zamaknili za eno urino periodo, tako da dobimo na izhodu pravilno vrednost. Realizacija poteka v treh nivojih:

- na najnižjem nivoju je MUX0 z vhodoma x0 in x1 ter izhod Y1
- na drugem nivoju je vmesna celica, ki prenaša signal x1
- na najvišjem nivoju pa signal x1, ki je vezan na izhod Y1

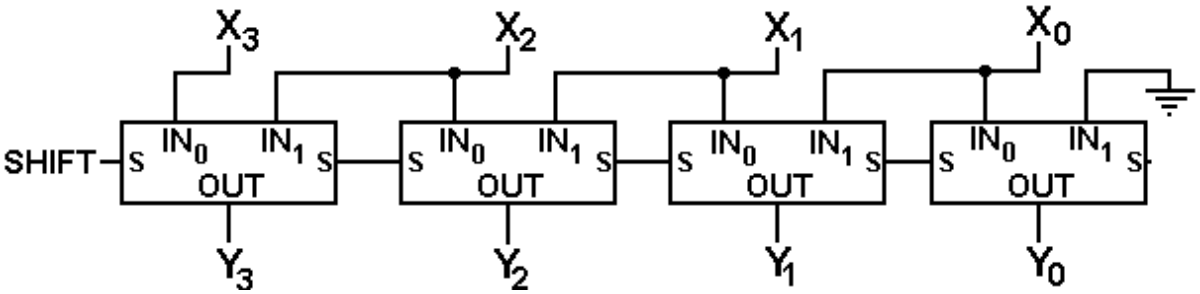


4-bitni pomikalniki

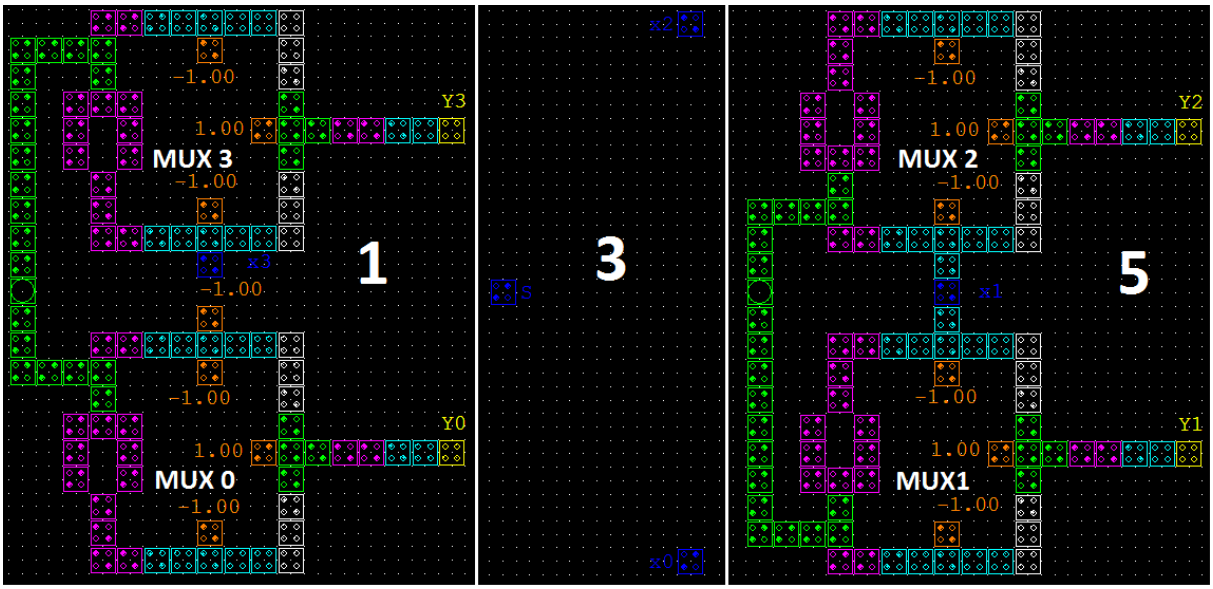
Realizacija 4-bitnih pomikalnikov je bolj zapletena od 2-bitne, saj tudi število potrebnih multiplekserjev naraste iz dveh na štiri pri pomiku za 0-1 mest in na osem pri pomiku za 0-3 mest. Kontrolni signal S pri pomiku za 0-1 mesto in kontrolna signala S0 in S1 pri pomikih za 0-3 mesta določajo število mest, za katere se premaknejo vhodni signali. Ti kontrolni signali so pripeljani na vhode multiplekserjev in določajo, kateri vhodi se povežejo na izhode.

Logični pomikalnik v levo za 0-1 bit

Najprej smo se osredotočili na logični pomikalnik v levo za 0-1 mesto, pri čemer smo uporabili kontrolni vhod S, štiri vhode x3-x0 in štiri izhode Y3-Y0.

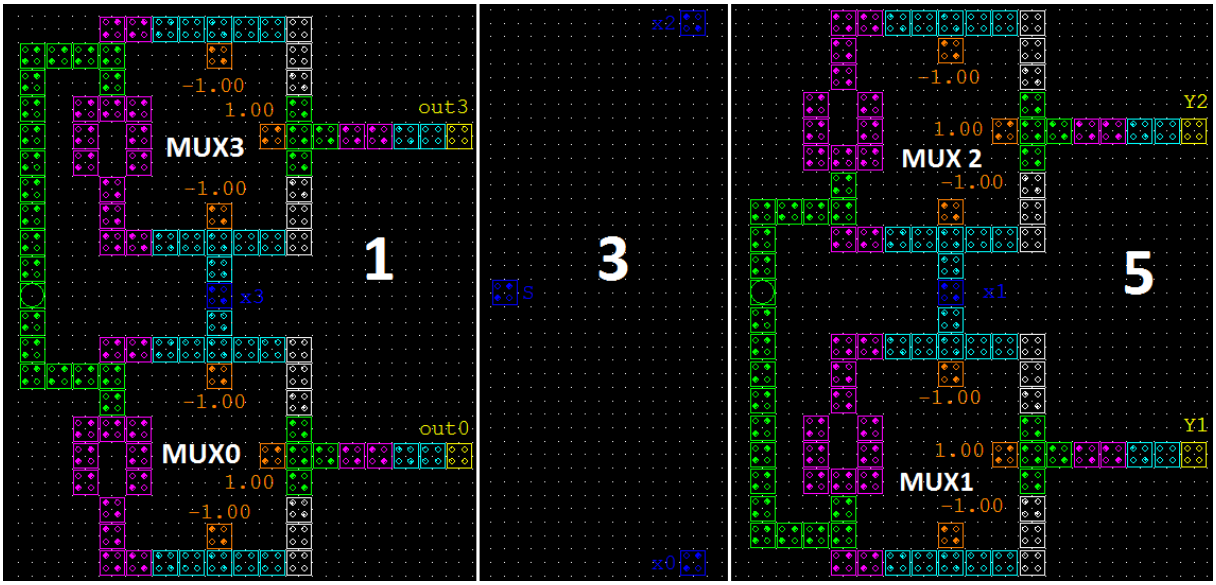


Zgornja shema nam je služila kot osnova, dejanska realizacija pa je bila precej drugačna. V QCA designerju smo naredili 5 plasti. Na prvi plasti sta multiplekserja, ki imata izhoda Y3 in Y0, na peti plasti pa multiplekserja z izhodoma Y2 in Y1. Na tretji plasti je kontrolni signal S in pa vhoda x2 in x0. Razlog za tako odločitev je v tem, ker so kontrolni signal S in vhoda x0 in x2 tako enako oddaljeni od prve in pete plasti, kar omogoča stabilnejše vezje. Na drugi in četrti plasti so samo celice, ki omogočajo prehode signalov x0 in x2 v mutliplekserje na plasteh 1 in 5.



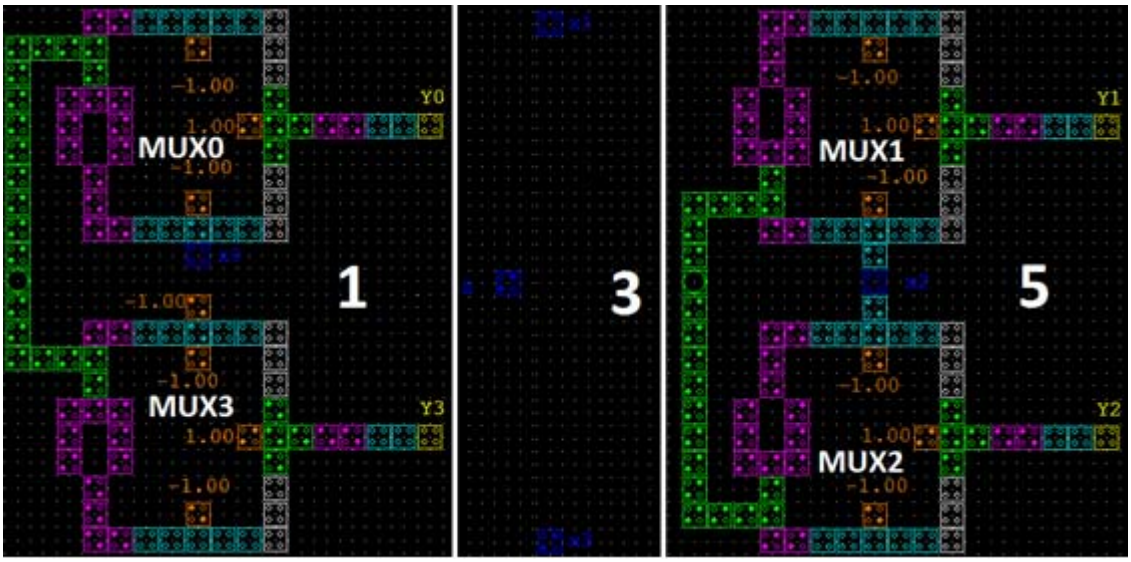
Krožni pomikalnik v levo za 0-1 bit

Krožni pomikalnik smo realizirali na skorajda identičen način kot logični pomikalnik v levo. Edina razlika je v tem, da je signal x3 na prvi plasti vhod tako v MUX0 kot v MUX3.



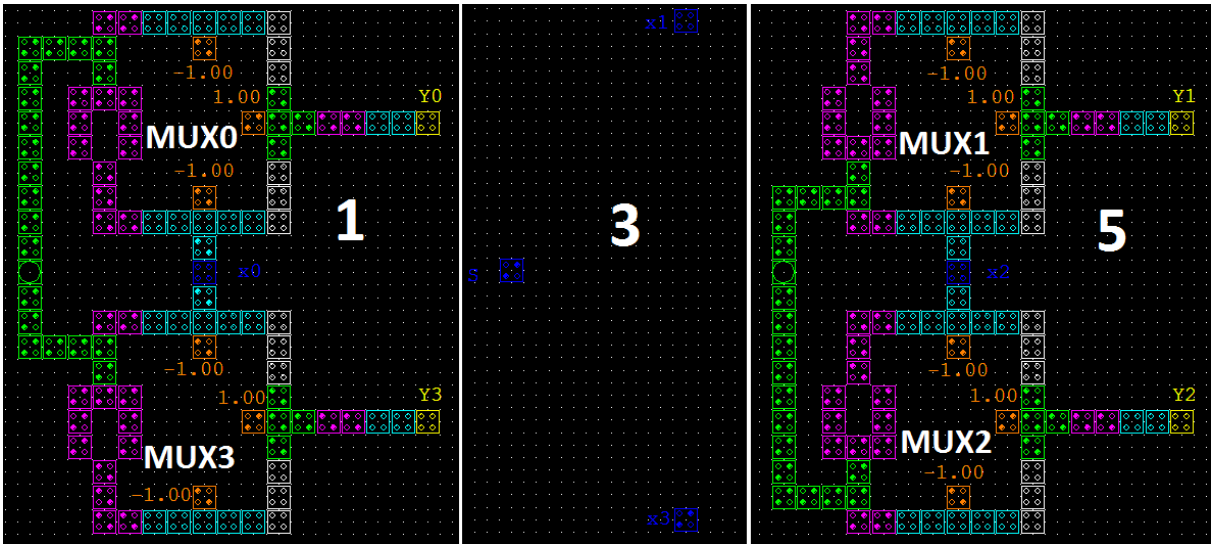
Logični pomikalnik v desno za 0-1 bit

Logični pomikalnik v desno je praktično zrcalna slika pomikalnika v levo. Med seboj zamenjamo vhode x0 in x3, ter x2 in x1. Prav tako med seboj zamenjamo izhode O0 in O3 ter O1 in O2.



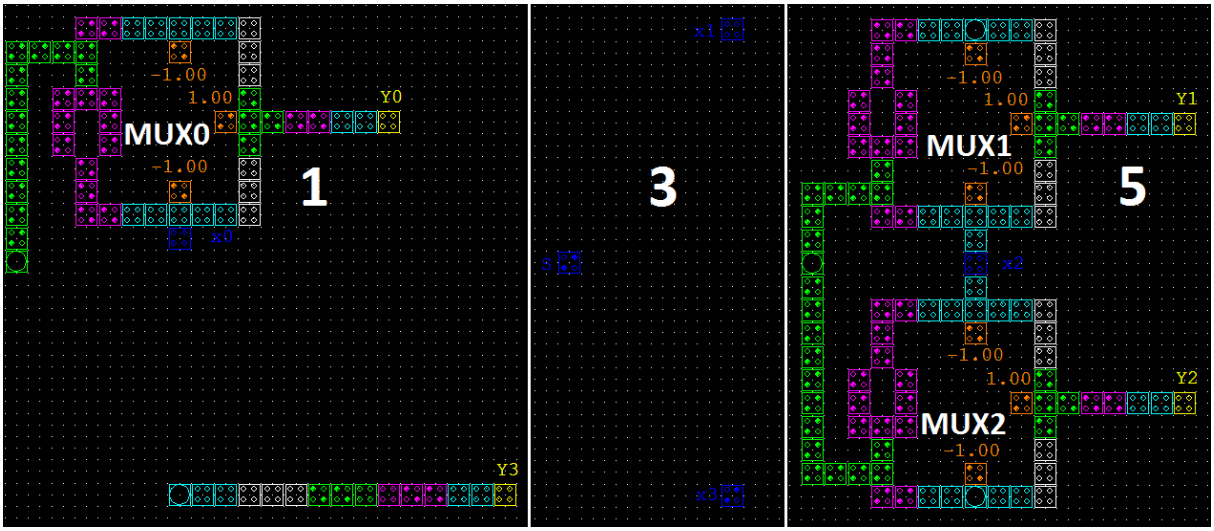
Krožni pomikalnik v desno za 0-1 mesto

Krožni pomikalnik v desno je zelo podoben logičnemu pomikalniku v desno. Razlikujeta se v vhodu v MUX3, kamor na vhod 1 pripeljemo x0, kar nam omogoča krožno delovanje pomikalnika.



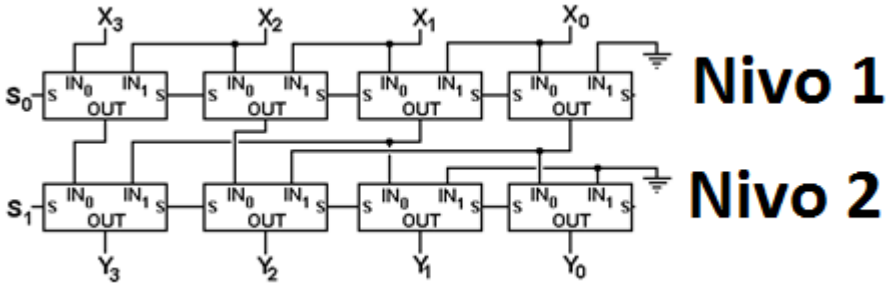
Aritmetični pomikalnik v desno za 0-1 mesto

Pri realizaciji aritmetičnega pomikalnika smo uporabili samo tri multipleksorje. Razlog za to je, ker se na izhod Y3 vedno prenese stanje vhoda x3, ne glede na aktivnost kontrolnega signala S.

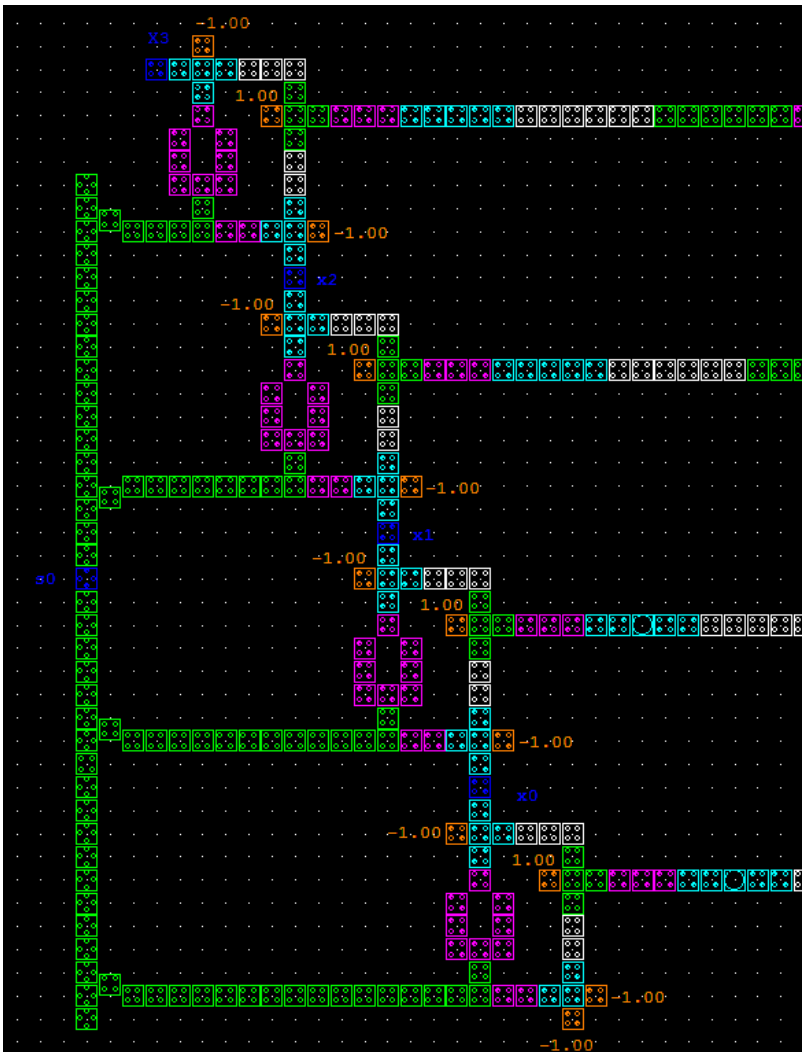


Logični pomikalnik v levo za 0-3 mesta

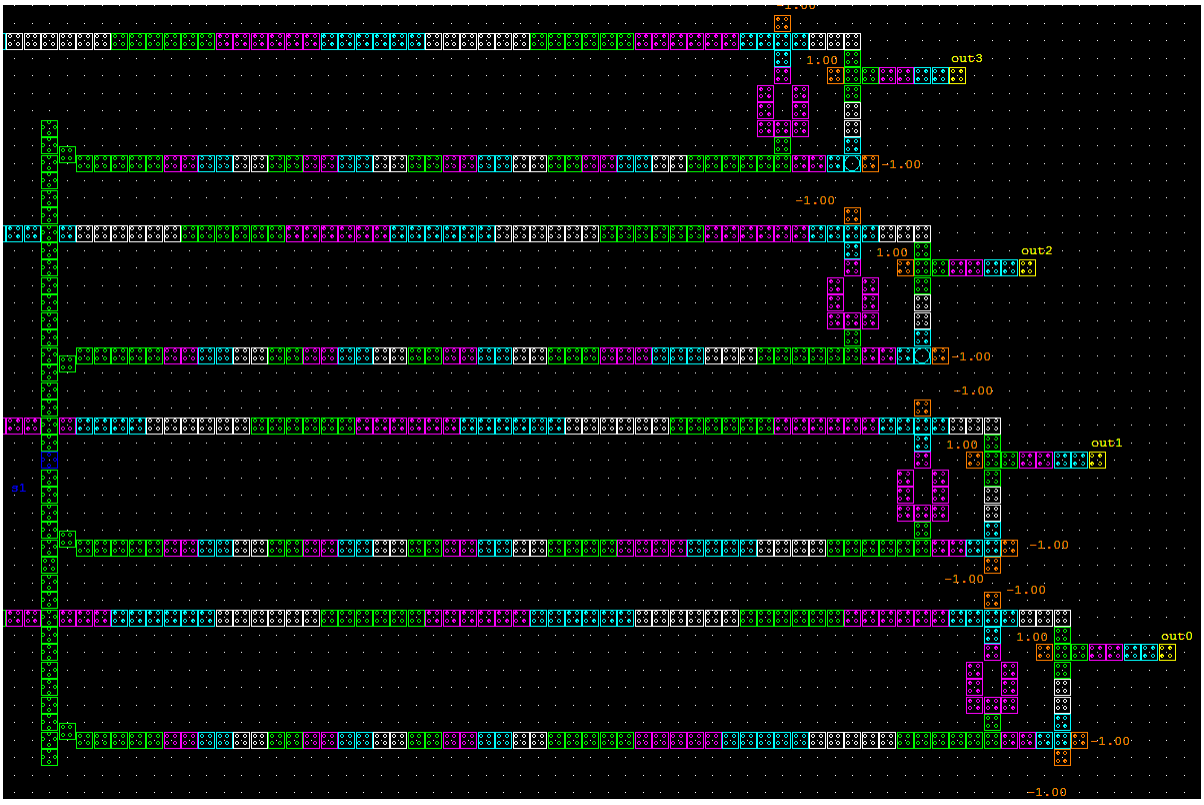
Realizacije pomikalnika smo se lotili na drugačen način kot ostalih. Pomikalnik smo naredili po spodnji shemi. Število multiplekserjev je osem, prav tako imamo dva kontrolna vhoda s0 in s1.



Prvi del sheme je sestavljen iz štirih multiplekserjev, ki naredijo pomik na prvem nivoju. Njihovo delovanje kontrolira signal s0, ki ga pripeljemo na vhode vseh MUX-ov.

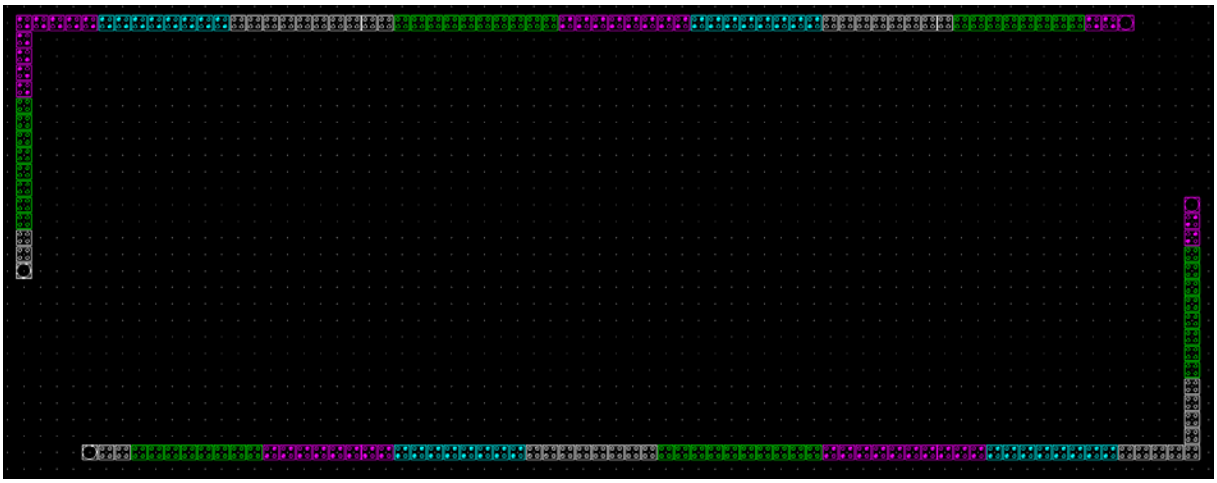


Zaradi večjega števila multiplekserjev in posledično povečanega števila celic smo imeli veliko težav z delovanjem, ker so postale celice in delovanje nestabilno. Težave smo odpravili tako, da smo vstavili tri vmesne urine periode.

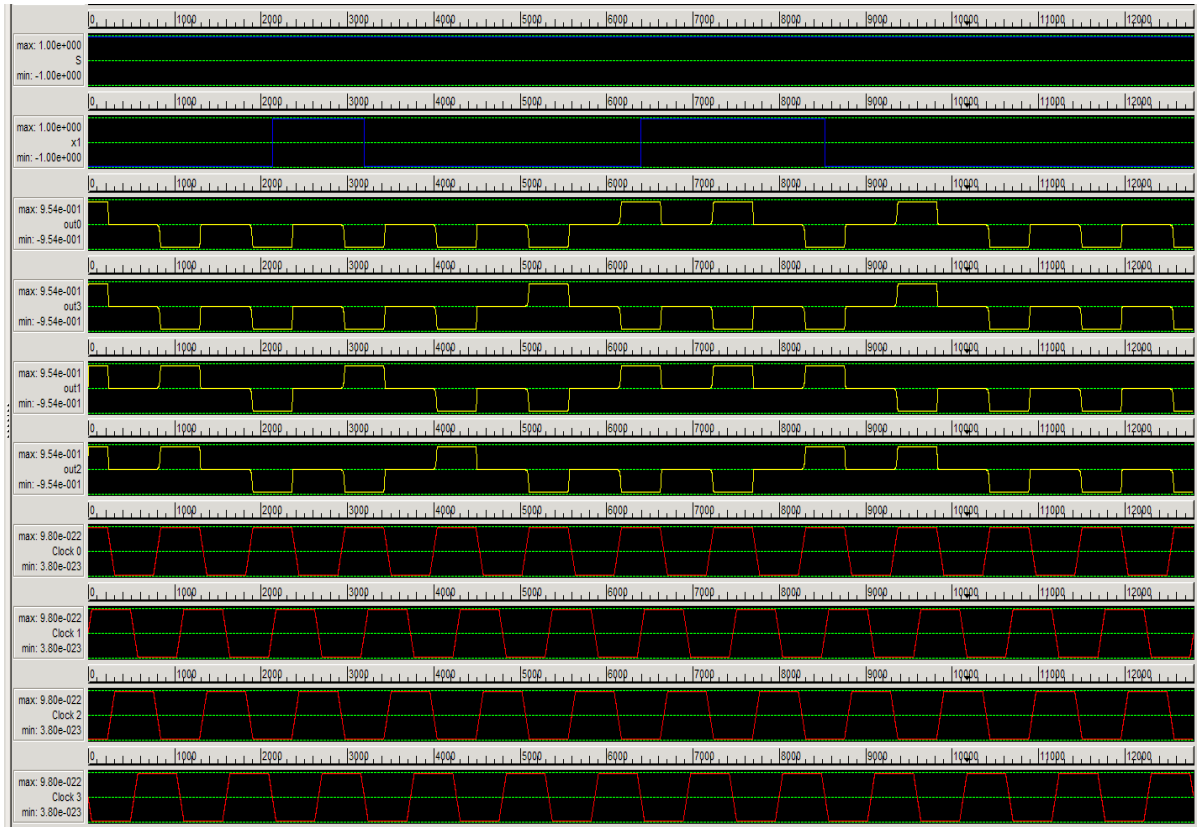


Drugi del sheme sestavljajo kontrolni signal s1 in štirje multiplekserji (nivo 2 na shemi), katerih vhodi so multiplekserji iz prvega nivoja.

Spodnjo strukturo smo uporabili za povezavo multiplekserjev 0 in 1 iz prvega nivoja z multiplekserji, ki imajo za izhod out2 in out1. Medsebojno vplivanje celic smo preprečili z vstavitvijo petih vmesnih nivojev.

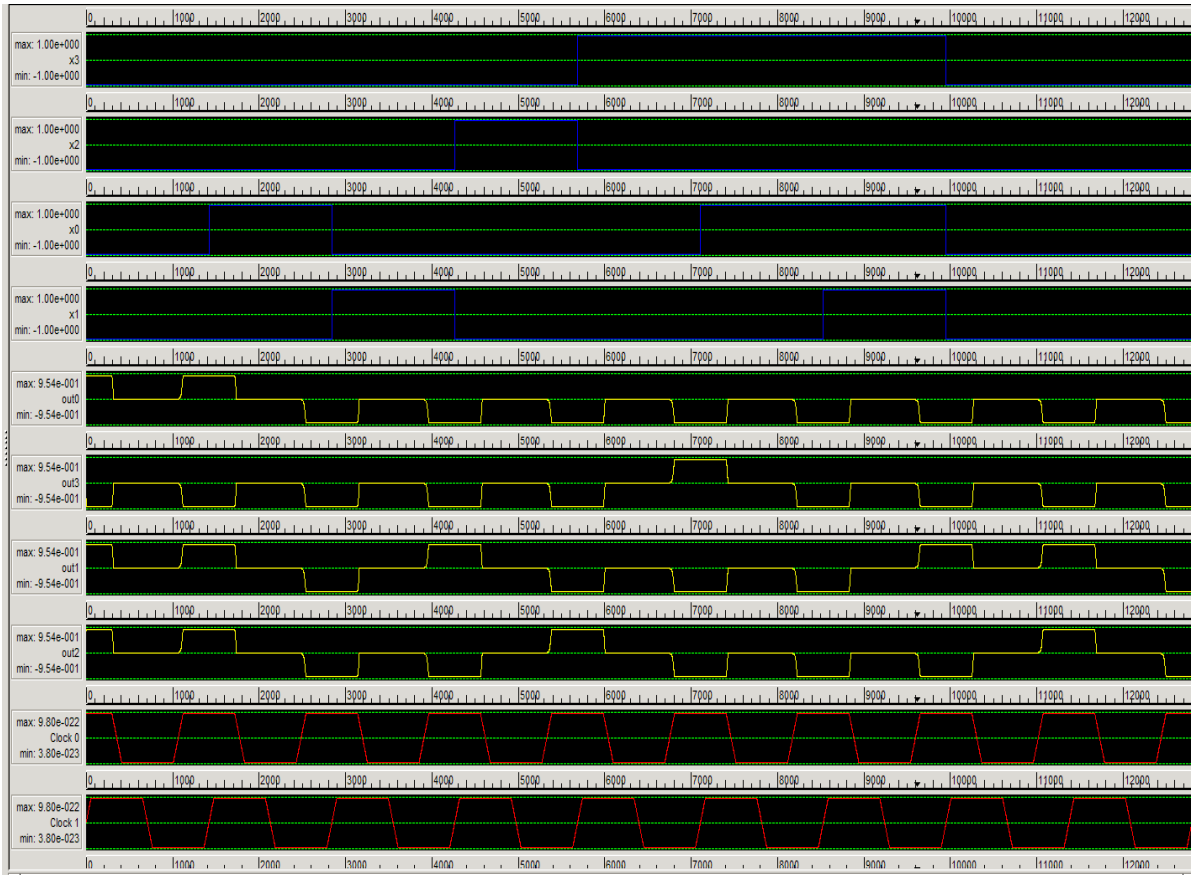


Rezultati



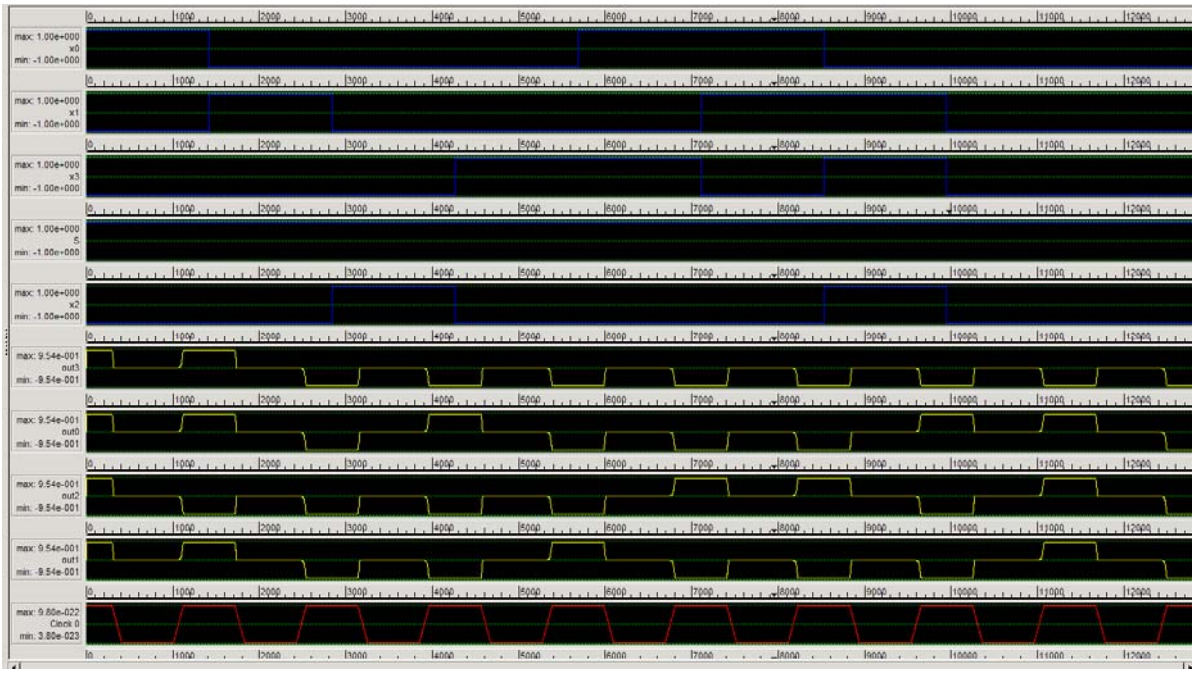
X0	1	0	0	0	1	1	0	0	0
X1	0	1	0	0	0	1	1	0	0
X2	0	0	1	0	0	0	1	0	0
X3	0	0	0	1	1	0	1	0	0
S	1	1	1	1	1	1	1	1	1
Out0	0	0	0	1	1	0	1	0	0
Out1	1	0	0	0	1	1	0	0	0
Out2	0	1	0	0	0	1	1	0	0
Out3	0	0	1	0	0	0	1	0	0

Slika 1: Rezultati pomika 4-bit krožnega pomikalnika v levo za eno mesto



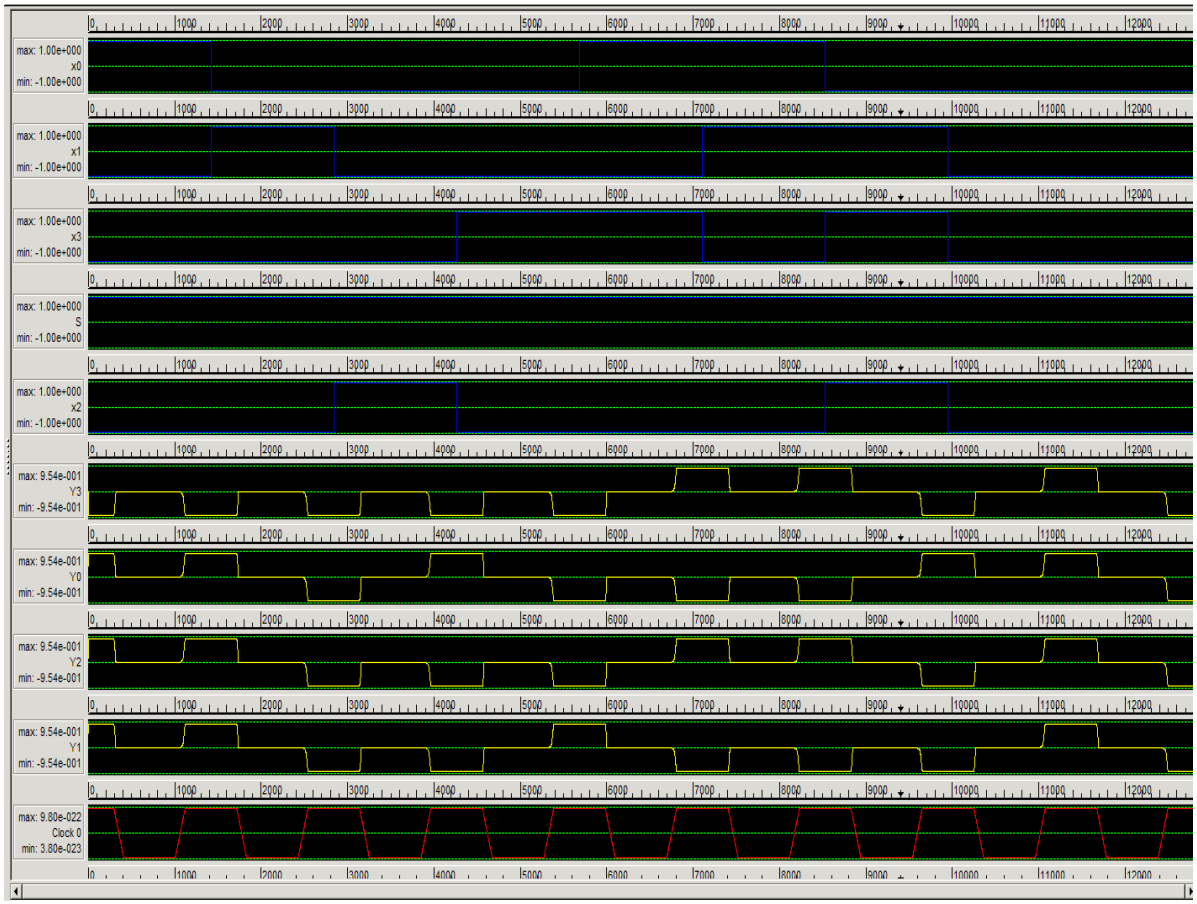
X0	1	0	0	0	1	1	0	0	0
X1	0	1	0	0	0	1	0	0	0
X2	0	0	1	0	0	0	0	0	0
X3	0	0	0	1	1	1	0	0	0
S	1	1	1	1	1	1	1	1	1
Out0	0	0	0	0	0	0	0	0	0
Out1	1	0	0	0	1	1	0	0	0
Out2	0	1	0	0	0	1	0	0	0
Out3	0	0	1	0	0	0	0	0	0

Slika 2: Rezultati pomika 4-bit logičnega pomikalnika v levo za eno mesto



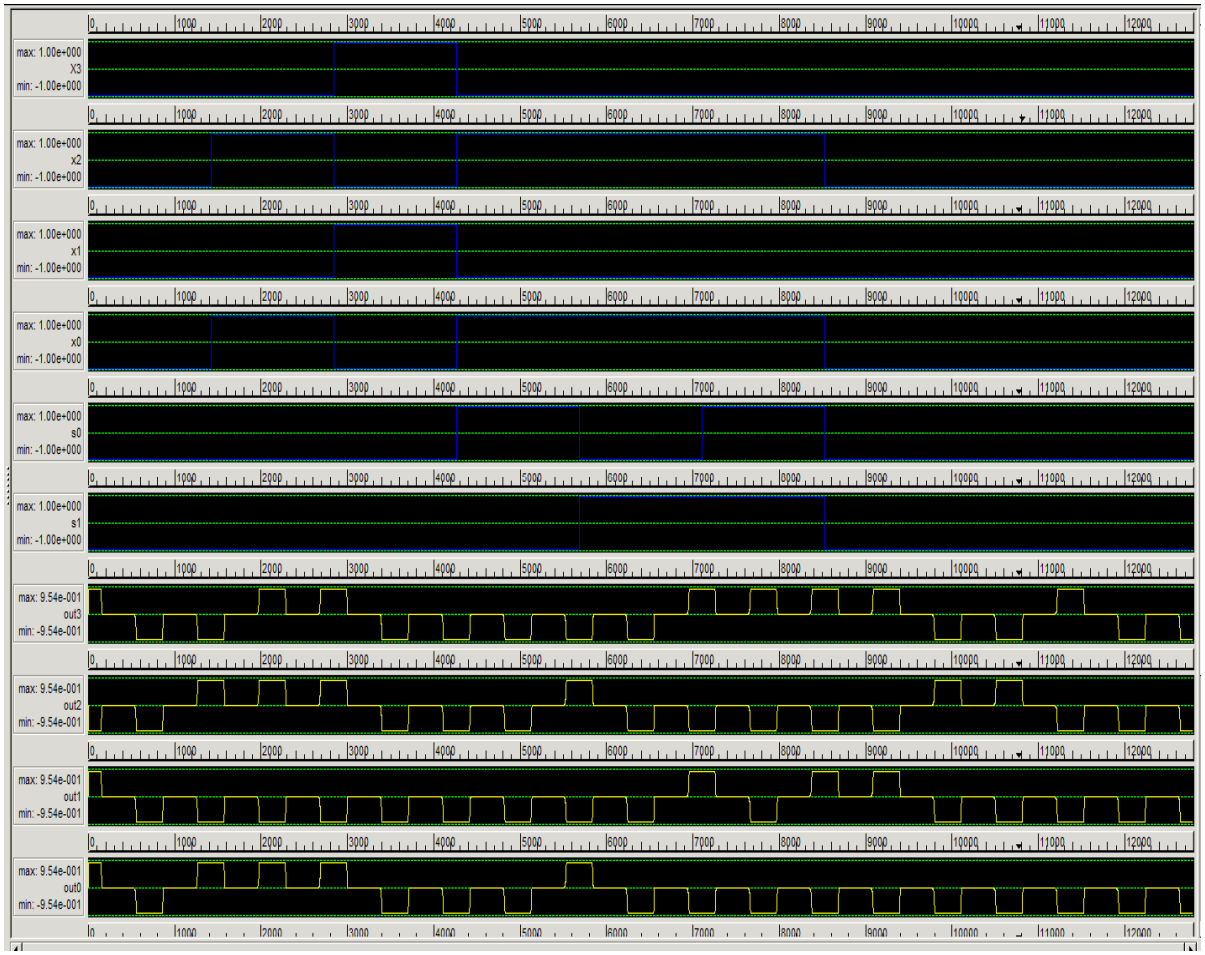
X0	1	0	0	0	1	1	0	0	0
X1	0	1	0	0	0	1	1	0	0
X2	0	0	1	0	0	0	1	0	0
X3	0	0	0	1	1	0	1	0	0
S	1	1	1	1	1	1	1	1	1
Out0	0	1	0	0	0	1	1	0	0
Out1	0	0	1	0	0	0	1	0	0
Out2	0	0	0	1	1	0	1	0	0
Out3	0	0	0	0	0	0	0	0	0

Slika 3: Rezultati pomika 4-bit logičnega pomikalnika v desno za eno mesto



X0	1	0	0	0	1	1	0	0	0
X1	0	1	0	0	0	1	1	0	0
X2	0	0	1	0	0	0	1	0	0
X3	0	0	0	1	1	0	1	0	0
S	1	1	1	1	1	1	1	1	1
Out0	0	1	0	0	0	1	1	0	0
Out1	0	0	1	0	0	0	1	0	0
Out2	0	0	0	1	1	0	1	0	0
Out3	0	0	0	1	1	0	1	0	0

Slika 4: Rezultati pomika 4-bit aritmetičnega pomikalnika v desno za eno mesto



X0	1	0	1	1	1	1	1	0	0
X1	0	1	0	0	0	1	0	0	0
X2	1	0	1	1	1	0	0	0	0
X3	0	1	0	0	0	0	0	0	0
S0	0	0	1	0	1	0	1	0	0
S1	0	0	0	1	1	1	1	0	0
Out0	1	0	0	0	0	0	0	0	0
Out1	0	1	1	0	0	1	0	0	0
Out2	1	0	0	1	0	1	0	0	0
Out3	0	1	1	0	1	0	1	0	0

Slika 5: Rezultati pomika 4-bit logičnega pomikalnika v levo za več mest

Zaključek

Pri delu smo imeli kar nekaj težav in večina je bila povezana s QCADesignerjem. Veliko časa nam je vzelo preslikovanje teoretične sheme vezja v QCA vezje. Ugotavljali smo, kaj je vzrok nedelovanja teoretično pravilnih struktur. Posledica tega je, da nam je za implementacijo vseh 4-bitnih pomikalnikov s to strategijo preprosto zmanjkalo časa. Ob predvidljivejšem delovanju QCADesignerja bi nam verjetno tudi to zagotovo uspelo. Smo pa ponosni na lepo urejeno, simetrično in majhno ter zanesljivo delujočo shemo.

Literatura

- Prof. Mraz, Miha. Predloge za predavanja pri predmetu ONT. Ljubljana: 2010
- http://en.wikipedia.org/wiki/Barrel_shifter
- http://web.cecs.pdx.edu/~mperkows/CLASS_FUTURE/QDCA/calgary-arithmetic-layout-ettc_qca.pdf