Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Tržaška 25

Ljubljana

Seminarska naloga

Kontrola križišča na osnovi števca in avtomata

Skupina 2

OPTIČNE IN NANOTEHOLOGIJE

Mentor: Primož Pečar

Ljubljana, 2.12.2010

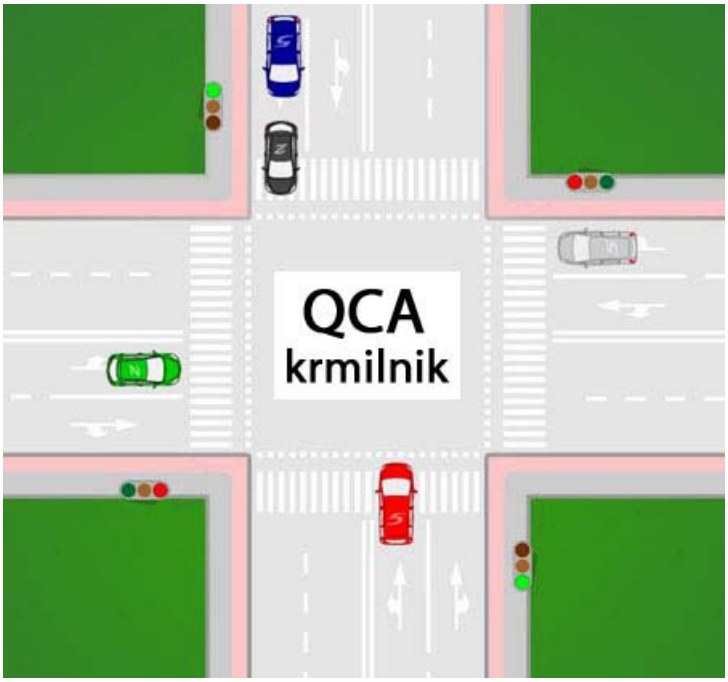
Jan Berdajs
Matevž Bizjak
David Lapajne
Marko Krečič

Kazalo

1. Uvod

Naša naloga je izvedba strukture v programu QCADesigner, ki kontrolira semaforizirano križišče – imamo torej 4 semaforje (po dva sta enaka). Za osnovo smo si vzeli delovanje semaforja, kot ga ponavadi srečamo na resničnih križiščih.

Ker je v naslovu naloge omenjena realizacija s števcem oz. avtomatom, smo morali stremeti k tej rešitvi naloge. Zamislili smo si delovanje, to pa zapisali v logično funkcijo, ki je primerna za uporabo s števcem. Sledila je minimizacija funkcij in ustrezno načrtovanje samega vezja s QCA strukturami.



2. Ideja in izvedba

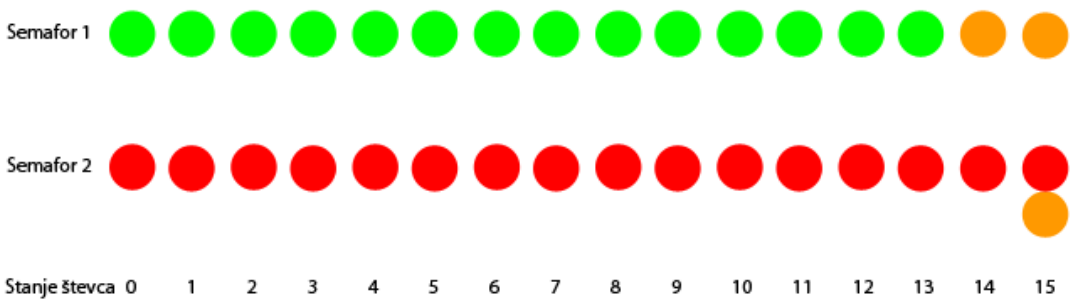
2.1 Logično vezje

Za realizacijo avtomata smo si zamislili števec. Števec šteje sinhronsko, v vsakem stanju števca pa so aktivirane določene luči na semaforjih. Ker imamo 4 semaforje – od tega imata po 2 enak način delovanja, smo poenostavili problem na 2 semaforja.

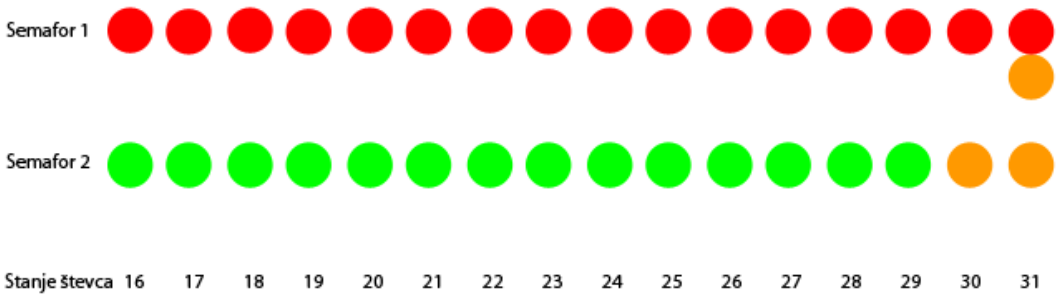
Odločili smo se, da traja en obhod avtomata **32 urinih period**. Za to je najprikladnejši 5 bitni števec. Delovanje semaforja smo razbili na urine periode, ki predstavljajo trenutno stanje na semaforju. Stanje luči pri določenem stanju števca je prikazano na slikah 1 in 2.

Prehodi luči:

- 14 period je semafor 1 zelen, semafor 2 je rdeč
- 1 periodo je semafor 1 oranžen, semafor 2 je še vedno rdeč
- 1 periodo je semafor 2 rdeč in oranžen hkrati (semafor 1 je še vedno oranžen)
- 14 period je semafor 1 rdeč, semafor 2 je zelen
- 1 periodo je semafor 2 oranžen, semafor 1 je še vedno rdeč
- 1 periodo je semafor 1 rdeč in oranžen hkrati (semafor 2 je še vedno oranžen)



Slika 1: Stanje semaforjev pri štetju od 0 do 15



Slika 2: Stanje semaforjev pri štetju od 16 do 31

Avtomat, ki deluje po tej shemi ima 6 izhodov:

- Semafor 1 zelen
- Semafor 1 oranžen

- Semafor 1 rdeč
- Semafor 2 zelen
- Semafor 2 oranžen
- Semafor 2 rdeč

Nato smo izdelali pravilnostno tabelo vseh teh izhodov našega avtomata (slika 3).

Števec	**Semafor 1**			**Semafor 2**		
	ZELEN	ORANZEN	RDEC	ZELEN	ORANZEN	RDEC
00000	1	0	0	0	0	1
00001	1	0	0	0	0	1
00010	1	0	0	0	0	1
00011	1	0	0	0	0	1
00100	1	0	0	0	0	1
00101	1	0	0	0	0	1
00110	1	0	0	0	0	1
00111	1	0	0	0	0	1
01000	1	0	0	0	0	1
01001	1	0	0	0	0	1
01010	1	0	0	0	0	1
01011	1	0	0	0	0	1
01100	1	0	0	0	0	1
01101	1	0	0	0	0	1
01110	0	1	0	0	0	1
01111	0	1	0	0	1	1
10000	0	0	1	1	0	0
10001	0	0	1	1	0	0
10010	0	0	1	1	0	0
10011	0	0	1	1	0	0
10100	0	0	1	1	0	0
10101	0	0	1	1	0	0
10110	0	0	1	1	0	0
10111	0	0	1	1	0	0
11000	0	0	1	1	0	0
11001	0	0	1	1	0	0
11010	0	0	1	1	0	0
11011	0	0	1	1	0	0
11100	0	0	1	1	0	0
11101	0	0	1	1	0	0
11110	0	0	1	0	1	0
11111	0	1	1	0	1	0

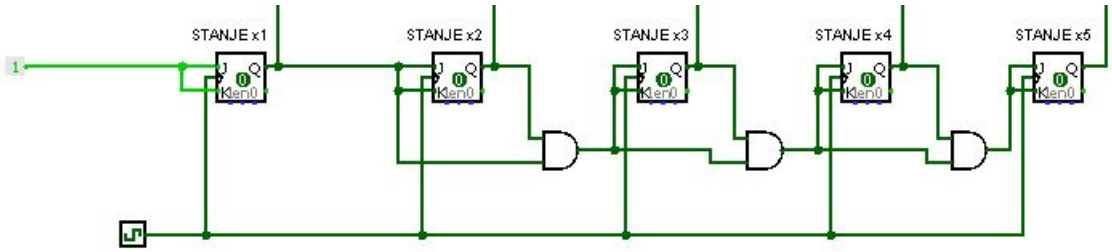
Slika 3: Pravilnostna tabela logične funkcije za izhod semaforjev

Ko smo imeli pravilnostno tabelo za vsakega izmed izhodov, je bilo potrebno izvesti še minimizacijo, da bi dobili čimbolj preprosto logično funkcijo. Za to smo uporabili **Karnaughov diagram**, ki je izboljšava Veitchevega diagrama. Uporabili smo aplikacijo, ki je navedena v literaturi – za vsak izhod smo vpisali želene vrednosti izhoda in kot rezultat dobili minimizirano logično funkcijo.

Minimizirane logične funkcije avtomata (x_5 MSB najbolj levi bit, x_1 LSB najbolj desni bit):

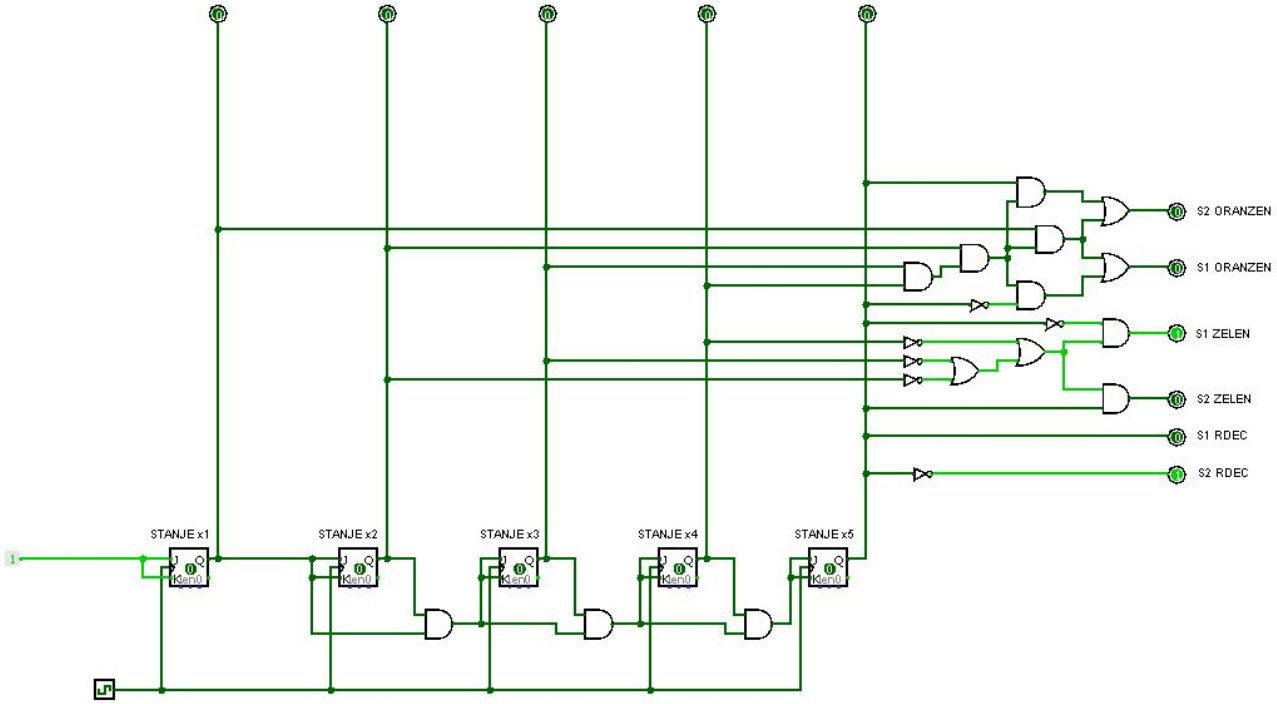
- Semafor 1 zelen: $\neg x_5 \wedge \neg x_2 + \neg x_5 \wedge \neg x_3 + \neg x_5 \wedge \neg x_4 = \neg x_5 \wedge (\neg x_2 + \neg x_3 + \neg x_4)$
- Semafor 1 oranžen: $x_4 \wedge x_3 \wedge x_2 \wedge x_1 + \neg x_5 \wedge x_4 \wedge x_3 \wedge x_2 = x_4 \wedge x_3 \wedge x_2 \wedge (\neg x_5 + x_1)$
- Semafor 1 rdeč: x_5
-
- Semafor 2 zelen: $x_5 \wedge \neg x_2 + x_5 \wedge \neg x_3 + x_5 \wedge \neg x_4 = x_5 \wedge (\neg x_2 + \neg x_3 + \neg x_4)$
- Semafor 2 oranžen: $x_4 \wedge x_3 \wedge x_2 \wedge x_1 + x_5 \wedge x_4 \wedge x_3 \wedge x_2 = x_4 \wedge x_3 \wedge x_2 \wedge (x_5 + x_1)$
- Semafor 2 rdeč: $\neg x_5$

Potrebovali smo le še 5-bitni sinhronski števec – realizirali smo ga z JK pomnilnimi celicami in AND vrati. Števec deluje tako, da se posamezen bit zamenja, ko so vsi manj pomembni biti v visokem logičnem stanju.



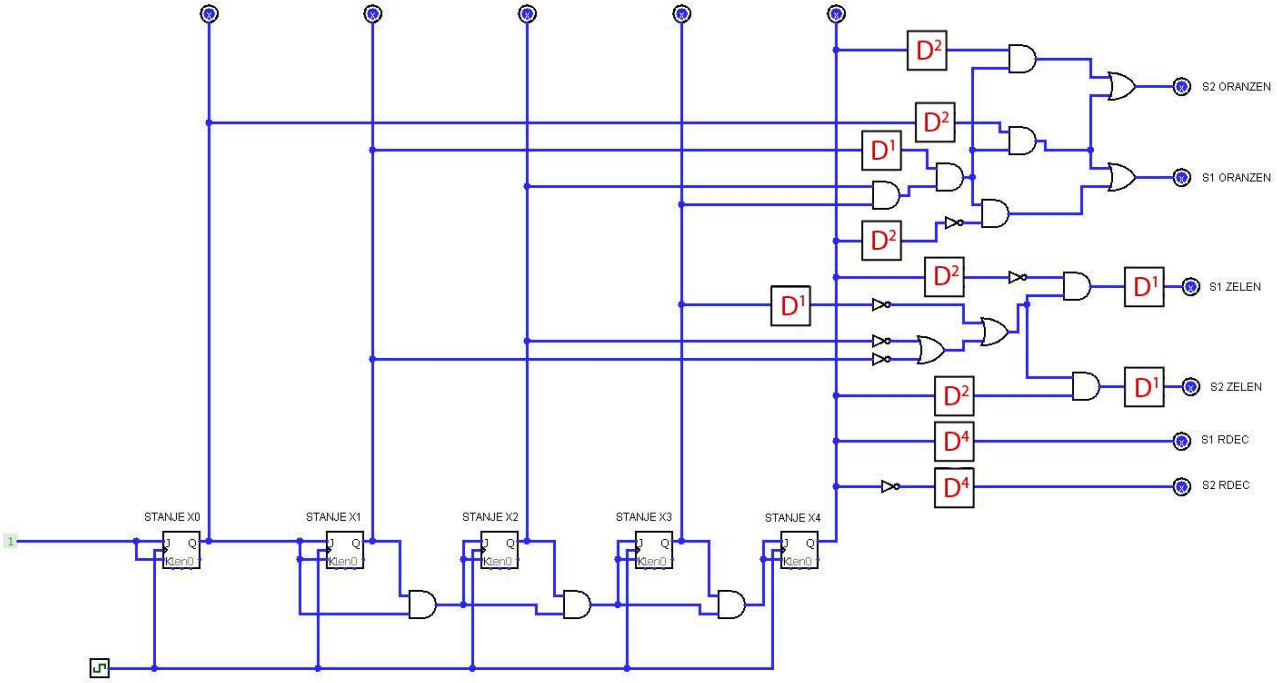
Slika 4: Logična shema 5-bitnega števca

Tako smo lahko sestavili celotno shemo našega vezja (slika 5) – izhodi števca so vhodi v nadaljnjo logiko, ki določa posamezne izhode.



Slika 5: Logična shema celotnega vezja

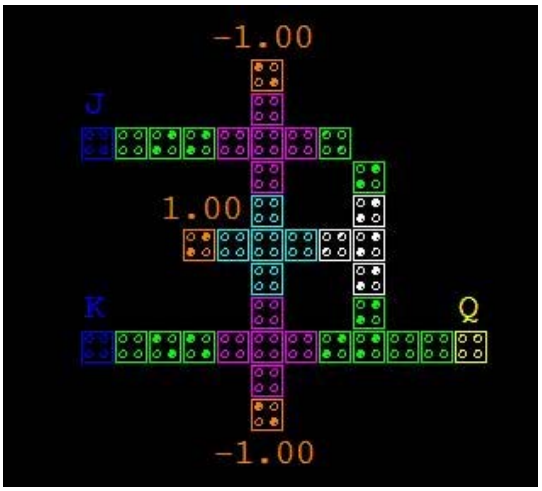
Potrebno je bilo tudi poskrbeti, da so signali v logiki na desni strani sheme sinhronizirani – zakasnitve, ki so potrebne za pravilno delovanje so prikazane na sliki 6.



Slika 6: Logična shema s potrebnimi zakasnitvami

2.2 Realizacija s QCA strukturami

Imeli smo pripravljeno osnovo, na podlagi katere je bilo potrebno izdelati še potrebne elemente s QCA strukturami ter jih pravilno povezati. Najprej smo izdelali delujočo JK pomnilno celico, ki je prikazana na sliki 7. S to celico smo nato realizirali 5-bitni seštevalnik.



Slika 7: JK pomnilna celica kot QCA struktura

Kot pri običajni CMOS JK pomnilni celici imamo vhoda J in K ter izhod Q. Vhoda za uro ne

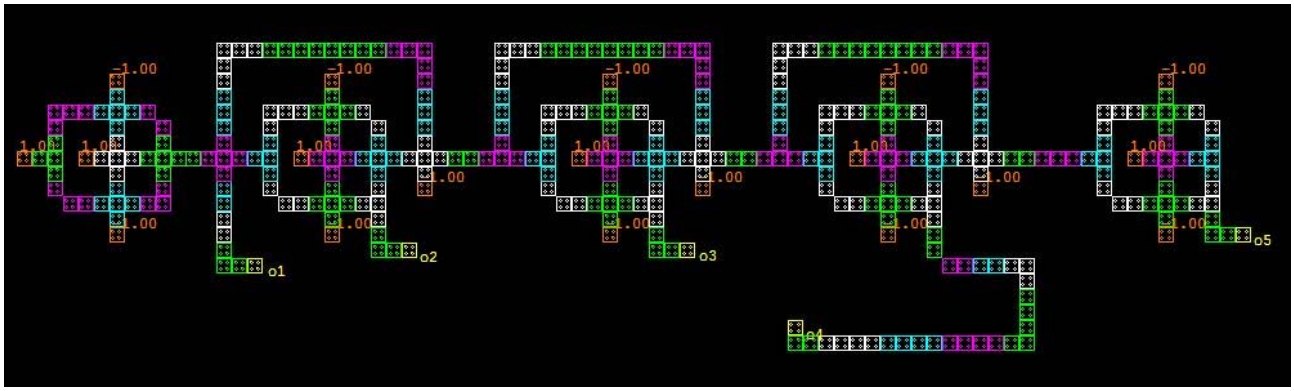
potrebujemo, saj že samo delovanje QCA struktur nudi sinhronizacijo z menjavanjem štirih faz (Switch, Hold, Release, Relax). Pomnjenje v celici je izvedeno z zanko, v kateri se informacija hrani. Naša JK celica je realizirana s tremi majoritetnimi vrati – dvoja opravljata funkcijo AND vrat, ena pa OR vrat.

Ko smo delovanje celice preizkusili smo lahko 5 JK celic povezali in pri tem uporabili še 3 AND vrata. Na tem mestu je bilo potrebno zagotoviti čim večjo simetričnost strukture, saj je zaradi tega struktura dosti bolj stabilna. Osnovna JK celica morda deluje, toda pri nepazljivem povezovanju večih celic na zelo nesimetričen način lahko povzroči nedelovanje celotnega števca. Na sliki 8 se vidi, da nam je števec uspelo realizirati relativno simetrično.

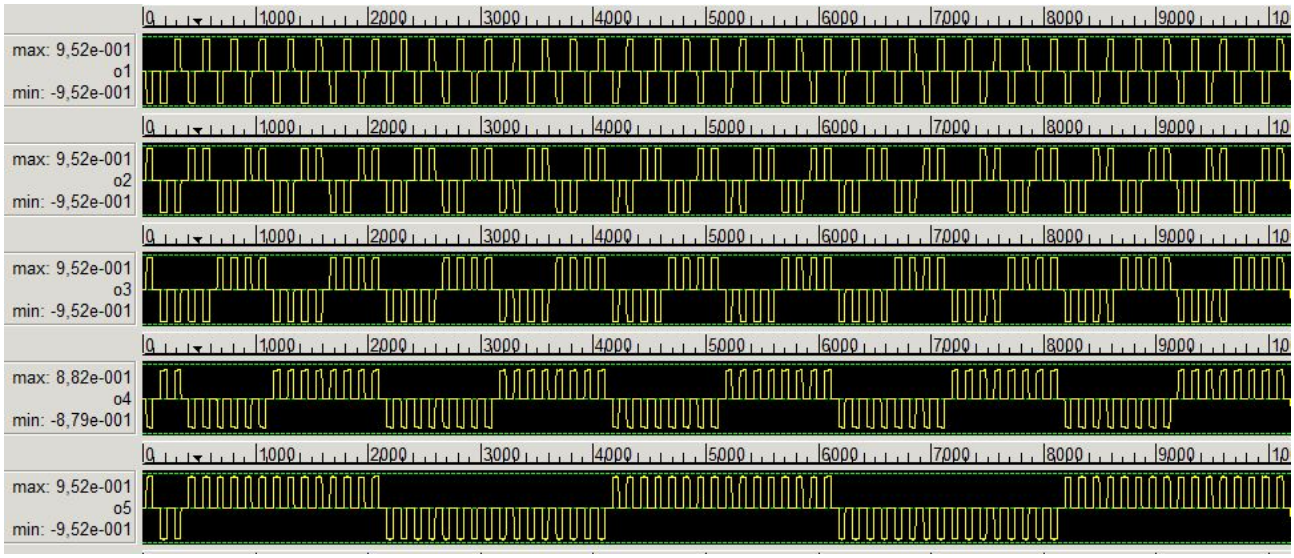
S tem smo izdelali delujoč 5-bitni števec, katerega izhode bomo uporabili za določanje pravilnih stanj semaforjev.

Na sliki 9 je vidno pravilno delovanje našega števca ob simulaciji, ne smemo upoštevati le nekaj začetnih period, ko simulacija še ne deluje pravilno. Najbolj leva celica števca ima za vhod logično enico, kar povzroči, da vsako periodo zamenja stanje – to predstavlja najnižji bit. Izhod te celice je na sliki 8 označen z o1. Na sliki je pri simulaciji predstavljenih vseh 5 izhodov števca in vidimo, da prva celica res vsako periodo zamenja stanje. Izhod te celice je pripeljan na vhod naslednje, pri tem pa z uporabo AND vrat zagotovimo, da se posamezni višji bit zamenja, ko so vsi manj pomembni biti v visokem logičnem stanju. Na sliki 9 vidimo, da to drži, saj izhod o2, ki predstavlja 2. bit res menja stanje v skladu s pravilnim delovanjem števca. To velja tudi za izhode o3, o4 in o5. Ko števec prešteje od 0 – vsi biti so postavljeni na 0 – do 31 – vsi biti števca so 1 – se vsi biti ponovno postavijo v vrednosti 0, tako pa števec ponovno ciklično šteje od 0 do 31. To nam koristi pri sami kontroli semaforjev, saj se tudi v realnosti semaforji ciklično ugašajo ter prižigajo.

Iz slike simulacije (slika 9) je razvidno tudi, da **so vsi izhodi števca stabilni** – prehodi med stanjema so hitri (nagel vzpon ter padec), nihanj v posameznem logičnem stanju pa ni (vidimo ravno linijo v posameznem stabilnem stanju.



Slika 8: 5-bitni števec na osnovi JK celic



Slika 9: Simulacija delovanja števca

3. Rezultati

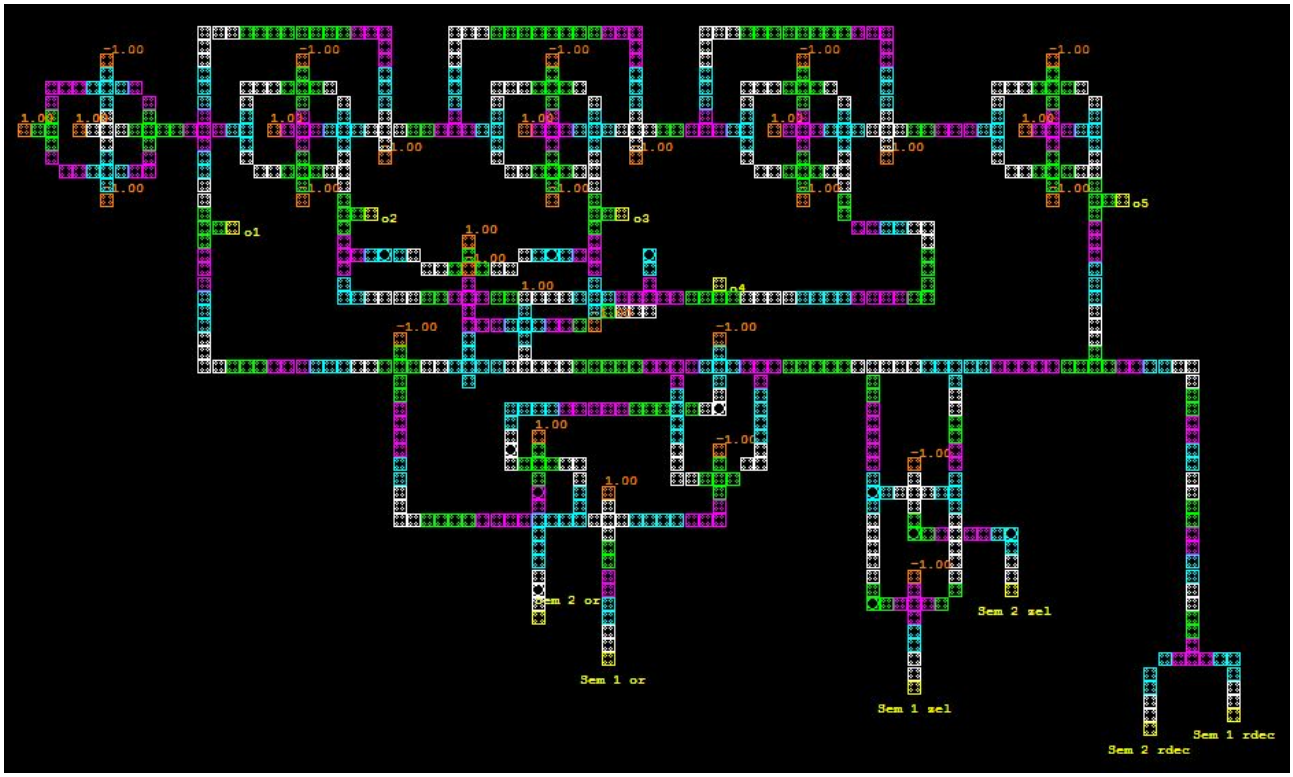
3.1 Izbrana realizacija

Celotna realizacija obsega **820 celic**, ki so porazdeljene na **3 nivoje**. Večina celic se nahaja na osnovnem nivoju, nekatere križne povezave pa so zaradi manjših motenj pri križanju speljane preko drugega nivoja na tretji nivo (en vmesni nivo je potreben zato, ker se ob prehodu na naslednji nivo signal negira, s še enim prehodom pa ima signal ponovno pravo vrednost). Srednji nivo torej vsebuje le skoznike.

Samo delovanje vezja je sledeče: na zgornjem delu se nahaja 5-bitni števec, katerega delovanje smo že razložili. Njegovih 5 izhodov pa smo nato speljali v logiko, ki smo jo dobili z minimizacijo pravilnostnih tabel za izhode posameznih luči obeh semaforjev.

V tej logiki je bilo uporabljenih le 11 majoritetnih vrat (7 AND vrat in 5 OR vrat) ter 6 osnovnih negatorjev - uporabili smo kar najpreprostejšo izvedbo le teh, saj nam niso povzročali problemov. V osnovi gre torej le za preprosto kombinatorično vezje, ki pa ne izgleda ravno najlepše, saj je povezovanje take logične funkcije v QCADesignerju zapleteno, ker je potrebno paziti tudi na ustrezne zakasnitve.

Zaradi preglednosti prilagamo tudi sliki posameznih plasti (slika 11 in 12).



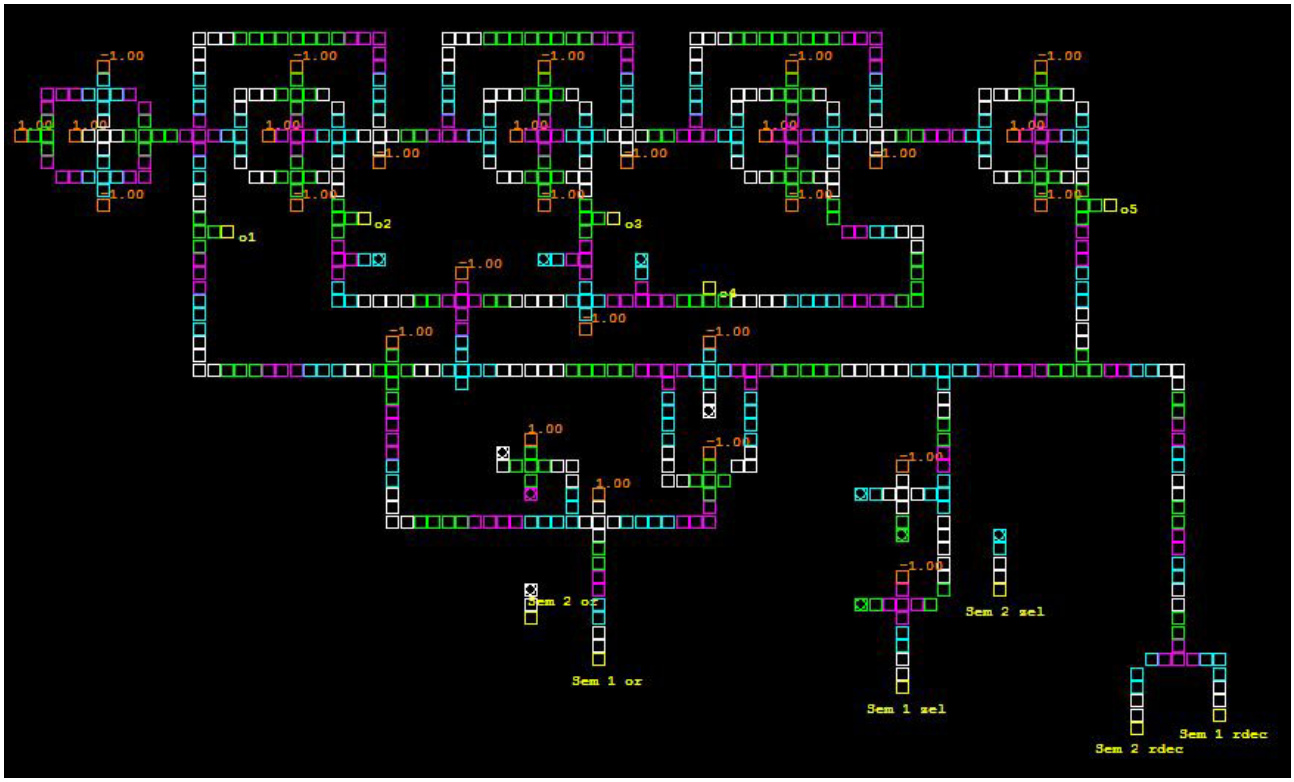
Slika 10: Realizacija avtomata za kontrolo križišča na 3 plasteh (vidne vse plasti)

Pri testnih izhodih o1-o5 so vsi izhodi števca sinhronizirani, saj je simulacija na sliki 9 pokazala, da števec deluje. Od tu naprej pa je potrebno poskrbeti, da bo tudi vseh 6 končnih izhodov sinhroniziranih oz. da bodo linije, ki vodijo do teh izhodov enako zakasnjene. S slike realizacije (slika 10) je razvidno, da so pri izhodih o1-o5 signali v fazi Switch (zelena barva). Od tu naprej se vsak signal do končnih izhodov **zakasni za 4 cele periode in za ¾ periode**, saj konča v fazi Relax

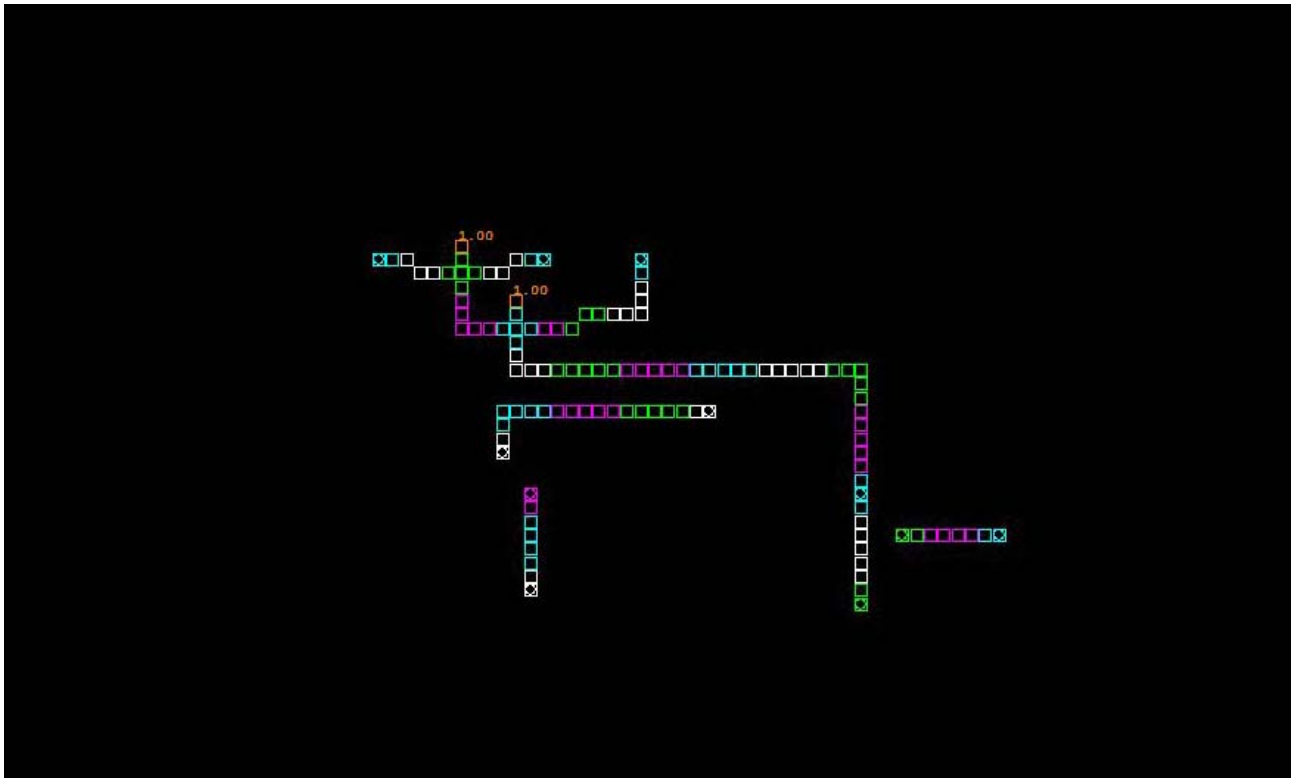
– bela barva. Samo povezovanje linij bi bilo možno izvesti na drugačen način in s tem priti do različne (tudi manjše) zakasnitve, vendar se s tem nismo preveč mučili, ker to na pravilnost delovanja vezja nima vpliva.

Na slikah 14 in 15 je opisano delovanje kombinatorične logike na koncu vezja na obeh plasteh – stanja $x_1 – x_5$ so izhodi števca. Za izhoda „Sem 1 rdec“ ter „Sem 2 rdec“ je potrebna logika trivialna – za prvega smo morali le prepeljati najvišji bit števca, za drugega pa negiran isti bit. Kot omenjeno, je bilo pri tem potrebno poskrbeti za zakasnitev 4 in ¾ peride. To je vidno na sliki 10, saj gre linija od testnega izhoda o5 do končnih izhodov „Sem 1 rdec“ ter „Sem 2 rdec“ štirikrat čez vse štiri faze ene periode, nato pa še čez 3 faze in konča v fazi Relax. Po istem principu je izvedena tudi logika za ostale 4 izhode, le da je tam situacija kompleksnejša, saj je potrebno uporabiti še majoritetna vrata in negatorje. Za lažjo predstavo smo na slikah 14 in 15 opisali potek in samo delovanje logike za ostale izhode.

Simulacijo števca lahko vidimo na sliki 9 - na začetku števec v večini primerov ne šteje od 0 dalje, saj se mora simulacija še inicializirati - celice se morajo pravilno postavit. Vezje začne pravilno delovati, ko števec začne šteti od 0 naprej - tu imamo še zakasnitev 4 ¾ periode, da dobimo želene izhode za luči semaforjev. Ta zakasnitev je z rdečo puščico označena na sliki 13. Na tej sliki tudi vidimo, da so z upoštevanjem zakasnitve izhodi za semaforje pravilni, se skladajo s pravilnostno tabelo (slika 3).

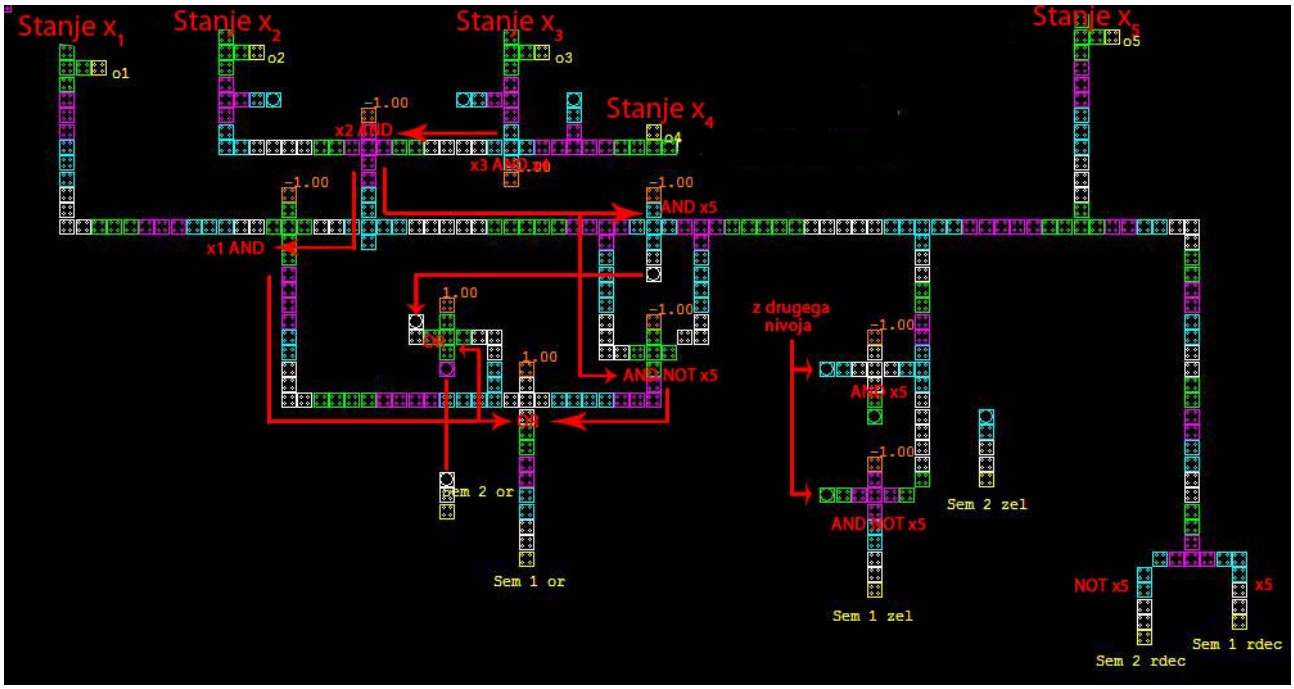


Slika 11: Vidna le najnižja plast

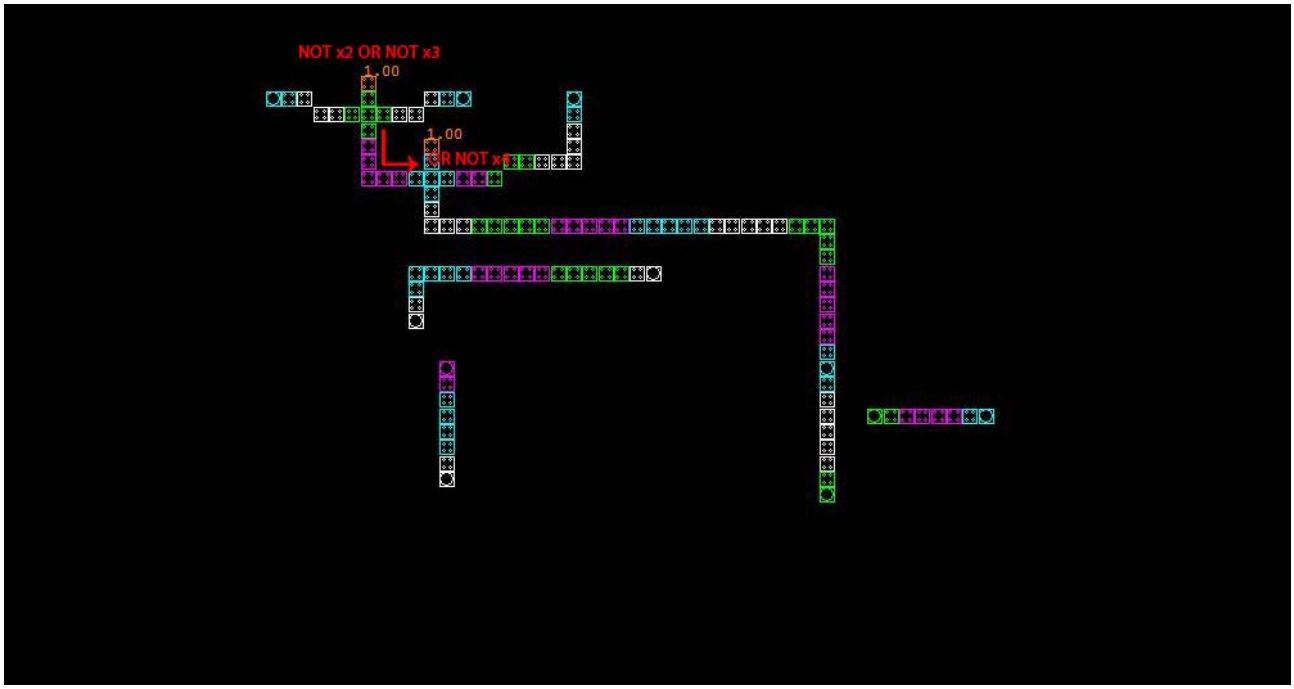


Slika 12: Vidna le najvišja plast

Slika 13: Rezultati simulacije



Slika 14: Logika na spodnji plasti

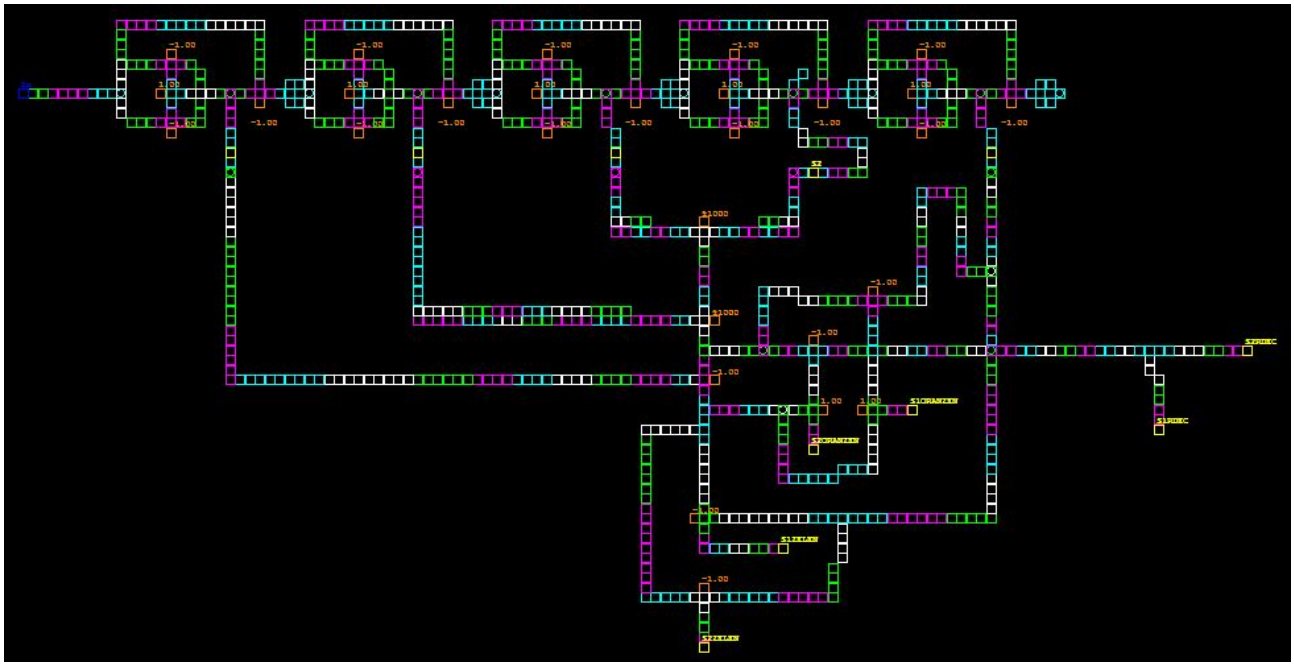


Slika 15: Logika na višjem nivoju

3.2 Alternativna realizacija

Kot omenjeno, izdelali smo še eno realizacijo, ki pa vsebuje več celic – 1035 in je realizirana na 12 nivojih. Števec je sicer realiziran na podoben način kot pri prejšnji različici, le da uporablja drugačne negatorje, povezovanje in končna logika pa sta izvedeni drugače. Ta realizacija je se izkazala za slabšo od zgoraj opisane zaradi števila porabljenih celic, poleg tega pa je bolj

nepregledna zaradi števila nivojev.



Slika 16: Alternativna realizacija (vidne vse plasti)

4. Zaključek

Na začetku smo imeli težave s samo realizacijo števca; ker imamo JK celice in AND vrata, vse te celice pa morajo biti sinhronizirane med seboj, je bilo treba premišljeno postavljati celice. Naredili smo več verzij števca, ki so tudi delovale, vse dokler nismo na izhode priklopili kombinatoričnega vezja - večkrat je števec en cikel štel pravilno, nato pa se je kakšna celica pokvarila in to je popolnoma zmotilo delovanje. Opazili smo, da imamo nestabilne linije in da sosednje celice vplivajo ena na drugo. Problem smo poskušali rešiti z več plastmi, toda tudi to ni bila popolna rešitev, saj so bili vplivi med celicami še vedno opazni. Na koncu smo za potrebe števca naredili dovolj simetrično JK celico, ki glede na rezultate simulacij vedno deluje pravilno in je zelo stabilna - nanjo lahko priklopimo nadaljnje vezje, brez da bi nam to zmotilo JK celice (števec).

Samo kombinatorično vezje je dokaj enostavno, morali smo paziti edino na sinhronizacijo ure, da so vsa majoritetna vrata dobila pravilne vhode. Nekaj težav smo imeli pri postavljanju linij, saj so za določene zakasnitve morale biti linije ustrezno dolge, spet v drugih primerih dovolj kratke zaradi stiske s prostorom. Nekaj povezav smo morali potegnili na drugo plast prav zaradi omejitev s prostorom, saj nam križanje linij ni uspelo.

Menimo, da so dodatne izboljšave možne, čeprav smo vezje skrčili na najmanjše število porabljenih celic. Mogoče bi se dalo še malo skrčit porabo celic, če bi števec naredili na več plasteh in bi izhode iz števca ustrezneje pripeljali do kompaktnejšega kombinatoričnega vezja.

5. Viri

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

http://www-ihs.theoinf.tu-ilmenau.de/~sane/projekte/karnaugh/embed_karnaugh.html