Kontrola križišča na osnovi števca in avtomata

Štirje semaforji

Seminarska naloga pri predmetu Optične in nanotehnologije

Avtorji: Pece Adzievski, Borče Paspalovski, Aleš Uršič, Andrej Babič

Mentor: Primož Pečar

Ljubljana, 30.11.2010

Kazalo

UVOD

Cilj seminarske naloge je bil izdelati avtomat za semaforizirano križišče s pomočjo kvantnih celičnih avtomatov. V križišču so štiri semaforji, po dva in dva imata vedno isto stanje. Prvotna zamisel monolitne implementacije se je izkazala za preveč omejujočo, zato smo jo izboljšali, in avtomat razdelili v dva ločena modula, spremenili pa smo tudi način delovanja (nove funkcionalnosti).

V seminarski nalogi so opisani in razloženi osnovni koncepti in podrobnosti implementacij obeh zamisli. Podrobno so predstavljeni rezultati simulacije delovanja in težave, s katerimi smo se srečali pri realizaciji naloge.

Uporabljena orodja

Pri izdelavi simulacije smo uporabili QCADesigner (glej vire).

Osnovni koncepti

Za bralčevo razumevanje vsebine seminarske naloge je potrebno vsaj osnovno znanje o konceptih kvantnih celičnih avtomatov in preklopnih vezij. Osnovni pojmi QCA (kvantnih celičnih avtomatov):

– **Kvantni celični avtomat (QCA)**
Kvantni celični avtomat je definiran kot dvo- ali več-stanjski sistem. Njegov osnovni gradnik je celica. Predstavljamo si ga lahko kot avtomat, ki izvaja določeno funkcijo.

– **QCA celica**
Osnovni gradnik vsakega kvantnega celičnega avtomata. Poznamo fiksno in osnovno celico, (delijo jih na podlagi tega, ali se njihova vrednost spreminja).

– **QCA struktura**
Več QCA celic združujemo v stukture, ki realizirajo logične funkcije. Z več celicami je mogoče realizirati tudi vidnik.

– **QCA ura**
Ura je realizirana kot električno polje nad gručo QCA celic. Ima štiri stanja, s katerimi nadzorujemo pretok podaktov. Skrbi tudi za napajanje.

Bralec si lahko več o QCA prebere iz na koncu navedenih virov.

IMPLEMENTACIJA SEMAFORJA

Realizirali smo dva avtomata, ker smo pri osnovni verziji opazili omejitev monolitne implementacije in pomanjkanje funkcionalnosti. Način realizacije nam ni omogočal enostavnega spreminjanja trajanja različnih stanj semaforja, (trajanje rdeče oz. zelene luči), in ni imel stanj, ki jih srečamo v obstoječih semaforjih, (npr. na vseh semaforjih prižgana rdeča luč, da se omogoči izpraznitev križišča).

Pri izboljšani verziji smo avtomat razdelili na dva modula. Prvi skrbi za logiko in stanja semaforja, medtem ko drugi generira čakalne periode.

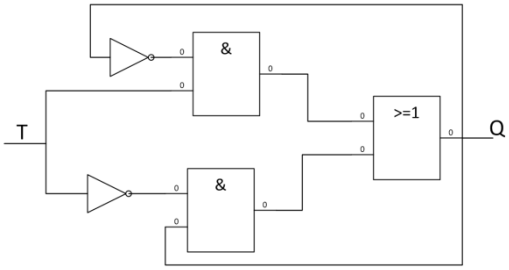
Poleg avtomata je bilo potrebno izdelati tudi kombinatorično vezje za priklop luči.

Ker gre pri problemu za križišče s štiri semaforji velja naslednja trditev: v vsakem trenutku imata dva in dva semaforja enako stanje. To nam omogoča, da v nadaljevanju besedila, (in v realizaciji avtomata), upoštevamo samo dva semaforja.

Osnovni gradnik obeh implementacij je T pomnilna celica, ki deluje po pravilnostni tabeli:

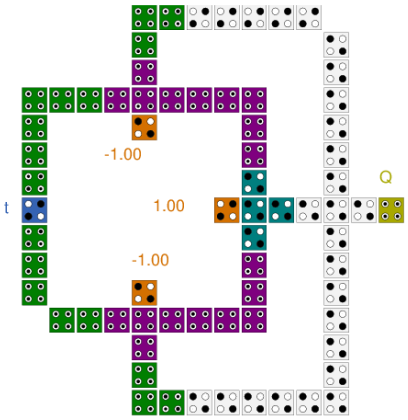
T(n)	Q(n)	Q(n+1)	Opis
0	0	0	Celica ohrani nizko stanje.
0	1	1	Celica ohrani visoko stanje.
1	0	1	Celica spremeni iz nizkega v visoko stanje.
1	1	0	Celica spremeni iz visokega na nizko stanje.

Logična shema T pomnilne celice:



Slika 1.1. Logična shema T pomnilne celice

Realizacija T pomnilne celice v QCADesigner-ju:



Slika 1.2. Realizacija T pomnilne celice v QCADesigner-ju

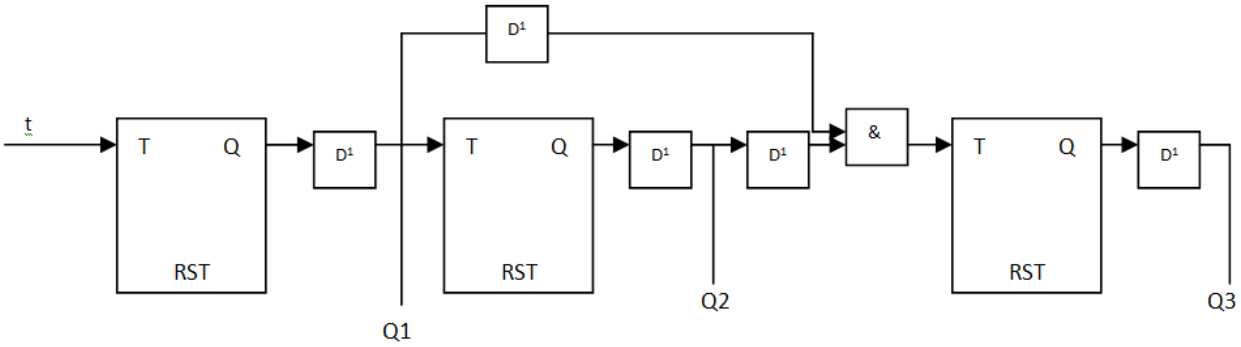
Osnovna monolitna implementacija

Koncept delovanja

Avtomat ima enostavno zgradbo in tri izhode, ki krmilijo prižiganje luči na semaforju, (potrebno je dodatno kombinatorično vezje). Ima tudi vhod, ki določa začetno stanje avtomata. Semafor ima samo osnovna stanja in ne predstavlja modela, kot ga srečamo na cestah.

Realizacija

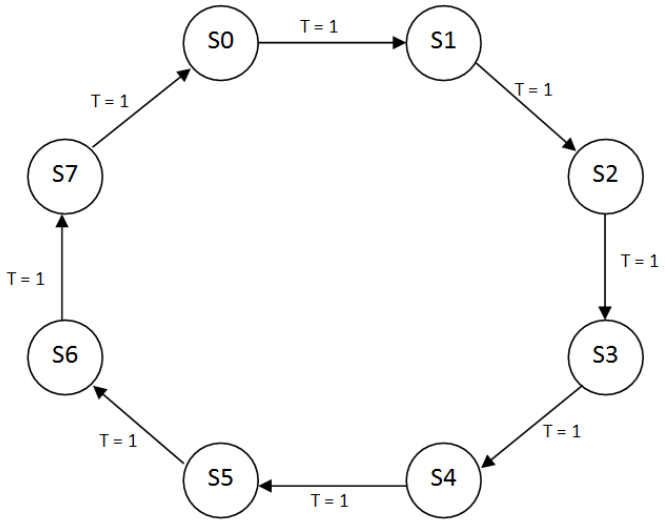
Avtomat je realiziran na podlagi treh T pomnilnih celic, ki krmilijo 3 izhode.



Slika 2. Logična shema avtomata, pri monolitni realizaciji

Izhodi Q1, Q2 in Q3 krimilijo kombinatorično vezje, ki prižiga in ugaša luči na obeh semaforjih.

Diagram prehajanja stanj avtomata:



Slika 3.Diagram prehajanja stanj avtomata pri monolitni realizaciji

Tabela stanj izhodov glede na stanje luči semaforjev:

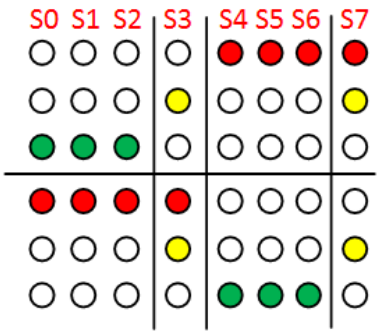
Stanje	Izhodi			Semafor 1			Semafor 2		
	Q3	**Q2**	**Q1**	**LR1**	**LRU1**	**LZ1**	**LR2**	**LRU2**	**LZ2**
S0	0	0	0	0	0	1	1	0	0
S1	0	0	1	0	0	1	1	0	0
S2	0	1	0	0	0	1	1	0	0
S3	0	1	1	0	1	0	1	1	0
S4	1	0	0	1	0	0	0	0	1
S5	1	0	1	1	0	0	0	0	1
S6	1	1	0	1	0	0	0	0	1
S7	1	1	1	1	1	0	0	1	0

LR# predstavlja stanje rdeče luči semaforja, LRU# predstavlja stanje rumene luči in LZ# predstavlja stanje zelene luči. Neposredno iz tabele izhajajo naslednje logične enačbe za krmiljenje semaforjev:

Semafor 1:
$LR1 = Q3$
$LRU1 = Q2 \wedge Q1$
$LZ1 = \neg Q3 \wedge \neg(Q2 \wedge Q1)$

Semafor 2:
$LR2 = \neg Q3$
$LRU2 = Q2 \wedge Q1$
$LZ2 = Q3 \wedge \neg(Q2 \wedge Q1)$

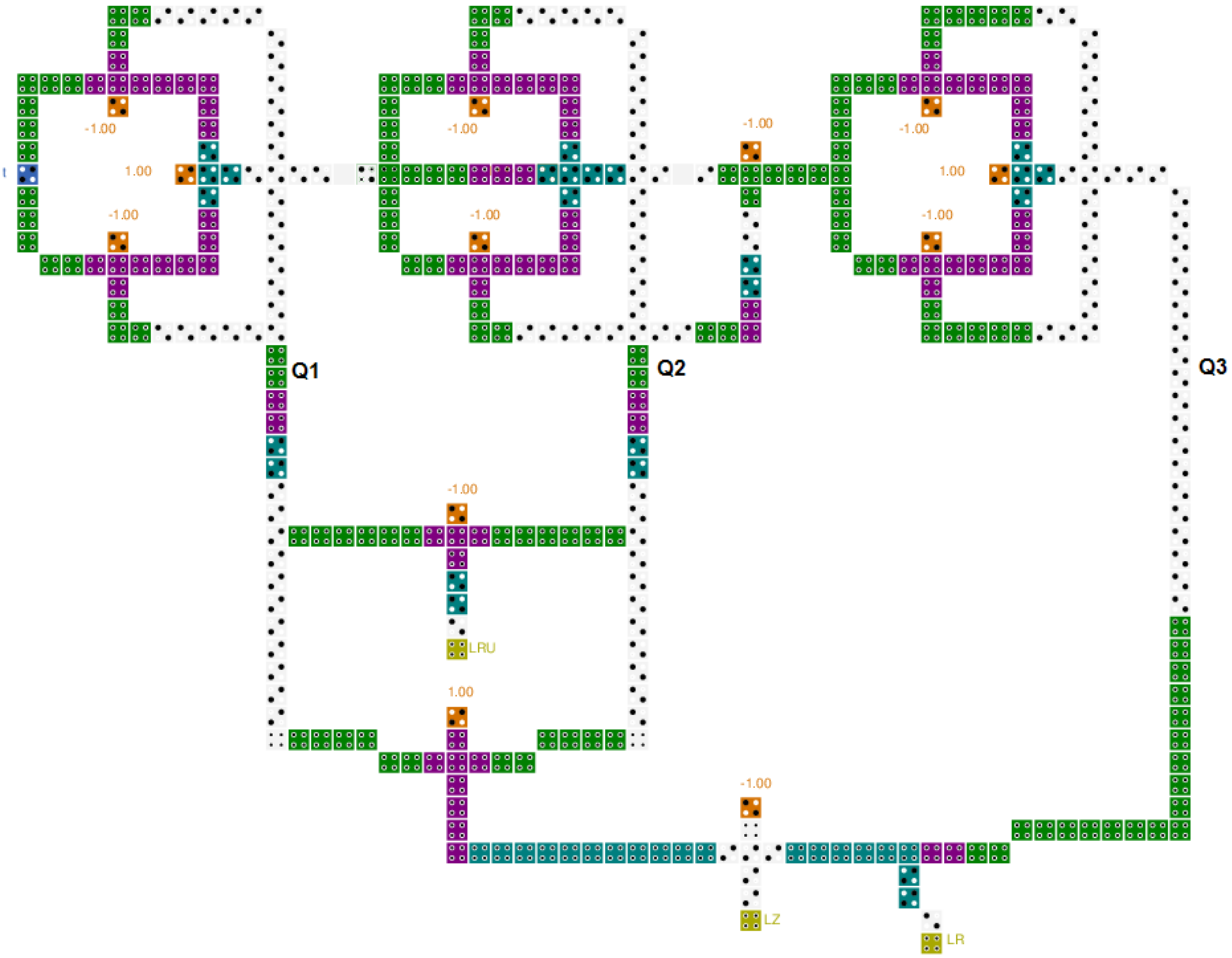
Iz diagrama in tabele prehajanja stanj lahko delovanje semaforja prikažemo grafično:



Slika 4.Grafična predstavitev posameznih stanj

Zelena luč traja tri urine periode, rdeča pa štiri. Razlika se pojavi zato, ker ostane rdeča luč prižgana tudi, ko je prižgana rumena luč (v prehodu med rdečo in zeleno lučjo).

Realizacija v programu QCADesigner:

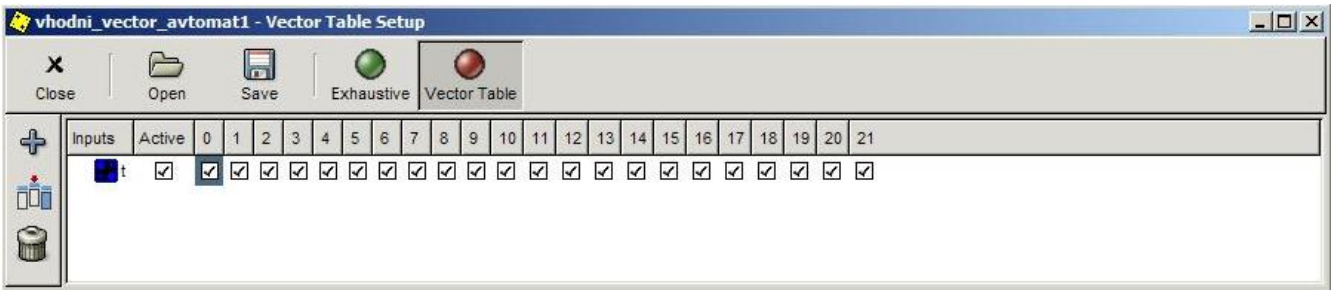


Slika 5.Realizacija monolitne implementacije v QCADesigner-ju

Rezultati

Avtomat deluje logično pravilno, vendar pa ne predstavlja realnega modela semaforjev, ki jih srečamo v vsakdanjem življenju. Manjkajo mu prehodna stanja med glavnimi stanji avtomata (zeleno / rdeče stanje). Z rešitvijo nismo bili zadovoljni, zato smo se odločili avtomat izboljšati.

Delovanje semaforja smo simulirali z testnim vektorjem:



Slika 6. Testni vektor za simulacijo avtomata

Opis delovanja

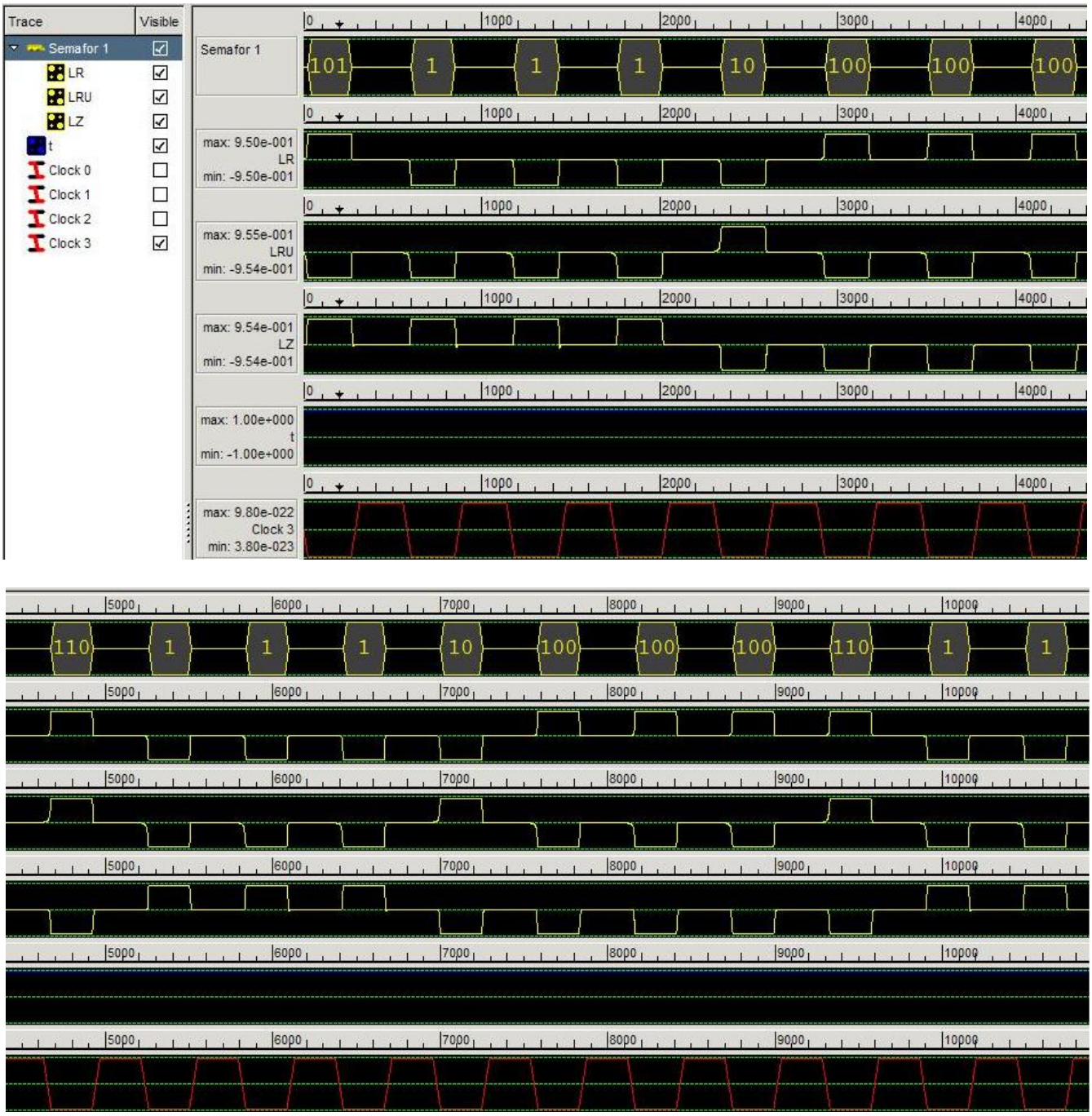
Osem notranjih stanj avtomata predstavlja štiri zunanja stanja semaforja: Rdeča, prehod na zeleno, zelena, prehod na rdečo:

– Rdeča luč traja tri urine periode (tri interna stanja).
– V prehodu med rdečo in zeleno sta istočasno prižgani rdeča in rumena luč, prehod traja eno urino periodo (eno interno stanje).
– Zelena luč traja tri urine periode (tri interna stanja).
– V prehodu med zeleno in rdečo je prižgana rumena luč. Prehod traja eno urino periodo (eno interno stanje).

Če seštejemo trajanje vseh stanj, dobimo, da je perioda semaforja osem. Oba semaforja imata isto zaporedje stanj, vendar je drugi semafor v primerjavi z prvim zamakjen za polovico periode (prvo stanje v drugem semaforju je peto stanje v prvemu).

Za bolj nazorni prikaz si bralec lahko ogleda rezultat simulacije na naslednji strani.

Rezultati simulacije:



Slika 7. Rezultati simulacije monolitne implementacije

Na sliki 7 je prikazana simulacija avtomata za kontrolo semaforja pri monolitni implementaciji. Vrednost (001) pomeni to, da na semaforju gori zelena luč, vrednost (100) rdeča luč, vrednost (010) rumena luč, vrednost (110) pa rdeča in rumena luč hlrato. Kot vidimo iz slike 7 zelena in rdeča luč trajata 3 urine periode, ostale pa po 1 urino periodo. Slika 7 na drugačen način prikazuje grafično predstavitev, ki jo vidimo na sliki 4.

Izboljšana modularna implementacija

Koncept delovanja

Avtomat je sestavljen iz dveh delov:

– **Števec za zamik**

Logičnemu delu avtomata podaja število čakalnih period za rdečo/zeleno luč. Ima vhod za definicijo začetnega stanja in vhod za ponastavljanje internega stanja.

– **Logični del**

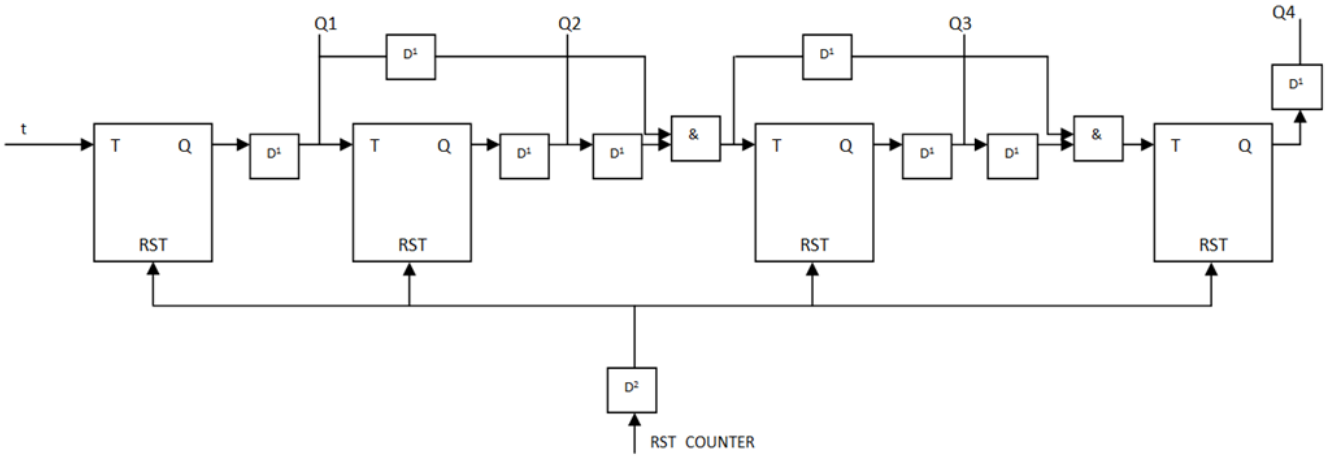
Je generator stanj semaforja, ki uporablja števec za zamik za določitev trajanja rdeče/zelene luči. Kot izhod podaja 3 spremenljivke, na podlagi katerih določimo stanje luči na semaforjih.

Realizacija

Ker sta realizaciji obeh delov neodvisni, jih bomo tako tudi obravnavali.

– **Števec za zamik**

Reliziran je kot zaporedje štirih T pomnilnih celic, kot je razvidno iz spodaj podane logične sheme.



Slika 8. Logična shema 4 bitnega števca

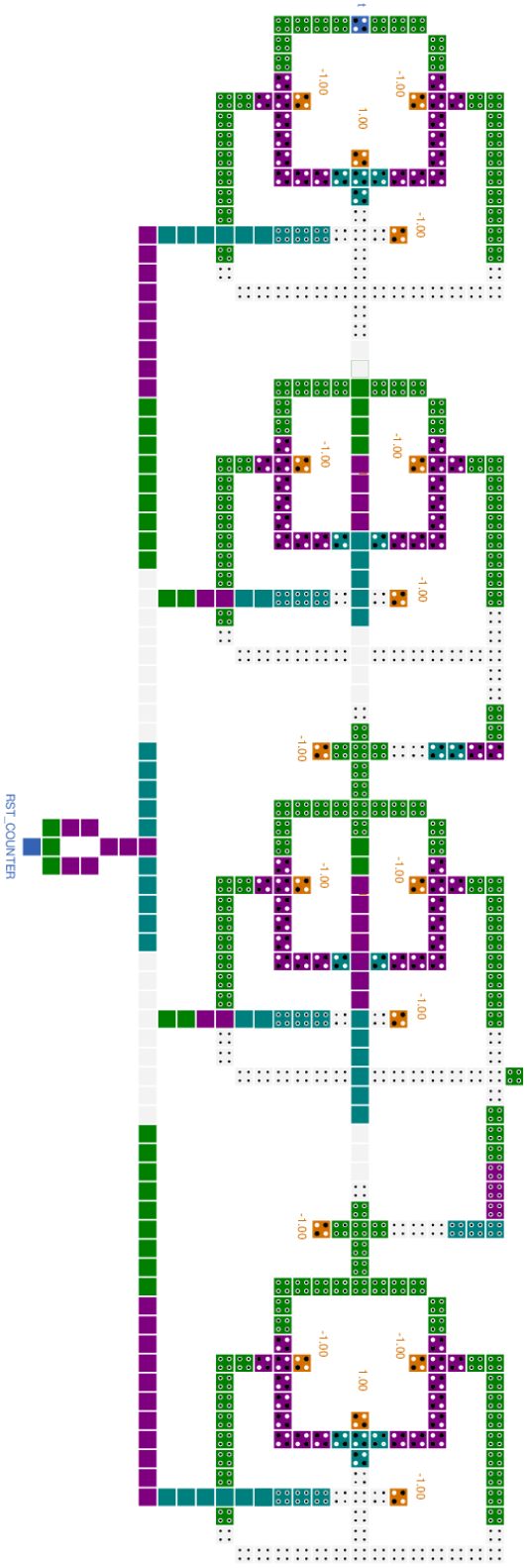
Vhod T omogoča štetje, vhod RST_COUNTER pa ponastavi stanje števca. Izhod se postavi v visoko stanje v četrti urini periodi, kar logičnemu delu avtomata poda zamik 4 urinih period.

Tabela prehajanja stanj števca glede na izhod v določeni urini periodi:

	T(0)	**T(1)**	**T(2)**	**T(3)**	**T(4)**	**T(5)**	**T(6)**	**T(7)**	**T(8)**	**T(9)**	**T(10)**	**T(11)**	**T(12)**	**T(13)**	**T(14)**	**T(15)**
Tr	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1

Kot je razvidno iz tabele števec spremeni stanje na izhodu vsako peto urino periodo (perioda števca je šestnajst). Za realizacijo števca v QCADesigner-ju smo uporabili 528 celic.

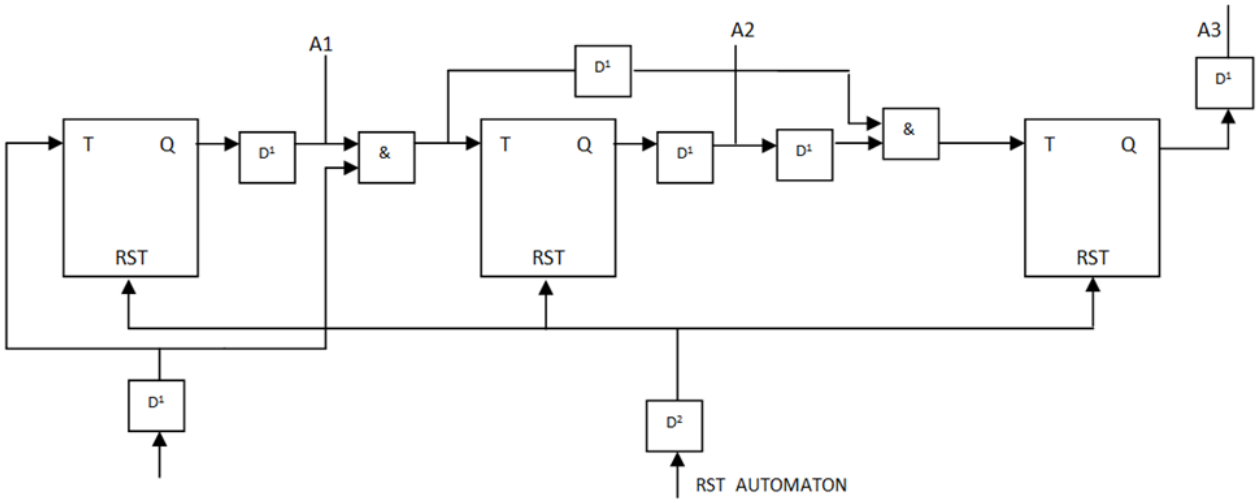
Realizacija števca v programu QCADesigner:



Slika 9. 4 bitni števec v QCADesigner-ju

– **Logični del**

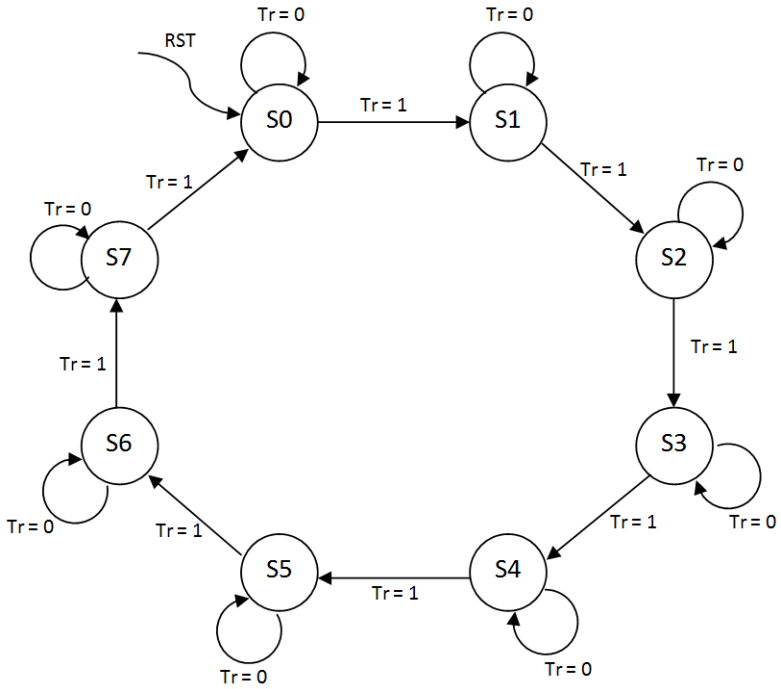
Reliziran je s tremi T pomnilnimi celicami, ki krmilijo tri izhode.



Slika 10. Logična shema avtomata, pri modularni realizaciji

Vhod STEVEC je izhod iz števca za zamik, vhod RST_AUTO pa ponastavi stanje logičnega dela avtomata. Izhodi A1, A2 in A3 tvorijo 8 različnih stanj, s katerimi krmilimo luči semaforjev.

Diagram prehajanja stanj avtomata:



Slika 11. Diagram prehajanja stanj avtomata pri modularni realizaciji

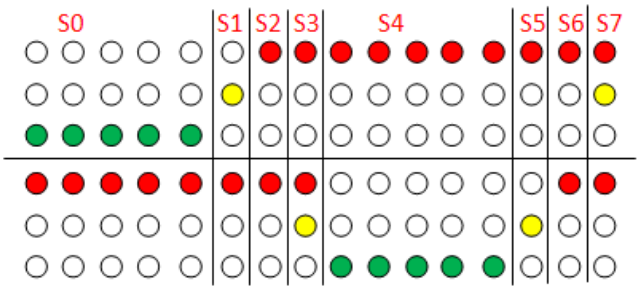
Tabela stanj izhodov A1, A2, A3 glede na stanje luči semaforjev:

Stanje	Izhodi			Semafor 1			Semafor 2		
	A3	**A2**	**A1**	**LR1**	**LRU1**	**LZ1**	**LR2**	**LRU2**	**LZ2**
S0	0	0	0	0	0	1	1	0	0
S1	0	0	1	0	1	0	1	0	0
S2	0	1	0	1	0	0	1	0	0
S3	0	1	1	1	0	0	1	1	0
S4	1	0	0	1	0	0	0	0	1
S5	1	0	1	1	0	0	0	1	0
S6	1	1	0	1	0	0	1	0	0
S7	1	1	1	1	1	0	1	0	0

LR# predstavlja stanje rdeče luči semaforja, LRU# predstavlja stanje rumene luči in LZ# predstavlja stanje zelene luči. Neposredno iz tabele izhajajo naslednje logične enačbe za krmiljenje semaforjev:

<u>Semafor 1:</u>
$LR_1 = A3 \vee A2$
$LRU_1 = (\neg A3 \wedge \neg A2 \wedge A1) \vee (A3 \wedge A2 \wedge A1)$
$LZ_1 = \neg A3 \wedge \neg A2 \wedge \neg A1$s
<u>Semafor 2:</u>
$LR2 = \neg A3 \vee A2$
$LRU2 = (\neg A3 \wedge A2 \wedge A1) \vee (A3 \wedge \neg A2 \wedge A1)$
$LZ2 = A3 \wedge \neg A2 \wedge \neg A1$

Iz diagrama in tabele prehajanja stanj lahko delovanje semaforja prikažemo grafično:

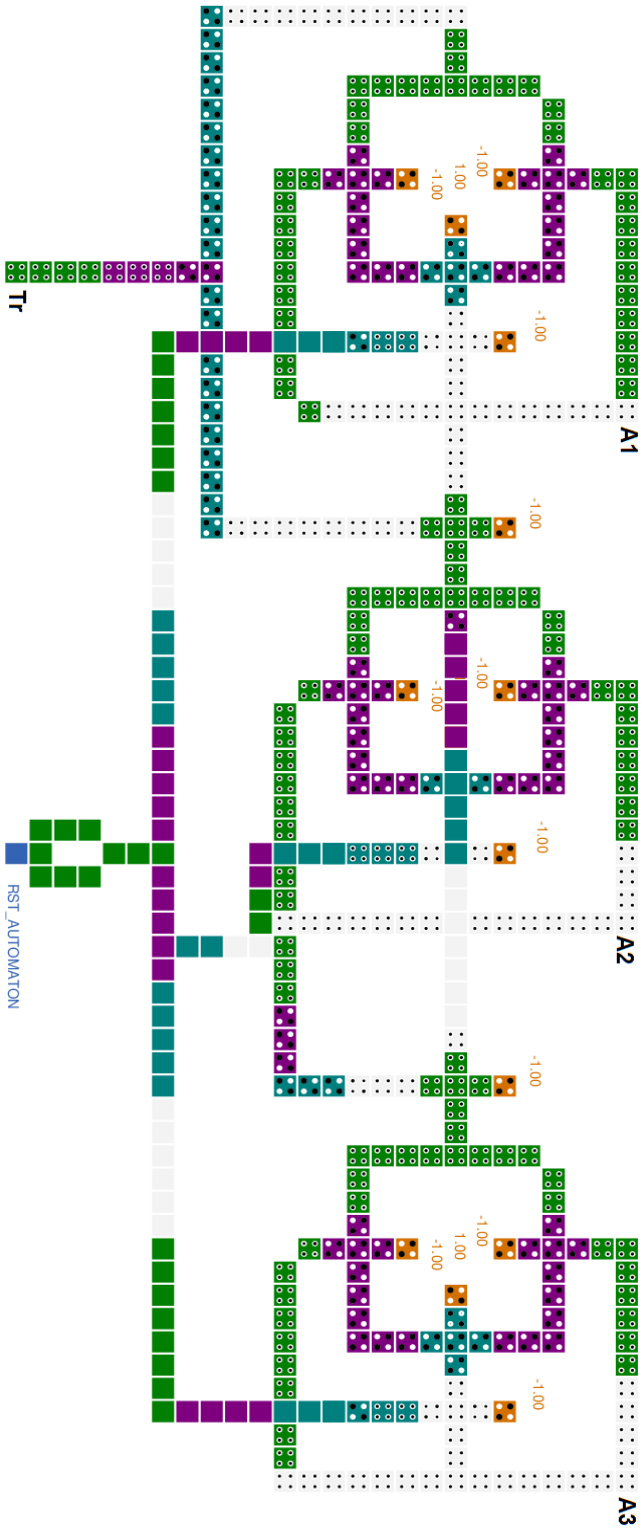


Slika 12. Grafična predstavitev posameznih stanj

Kot vidimo je trajanje zelene luči pet urinih period, medtem ko je trajanje rdeče luči deset urinih period. Razlika petih urinih period se pojavi iz treh razlogov:

– Potrebno je stanje, ko so vsi semaforji v križišču rdeči (praznjenje križišča).
– Med zeleno in rdečo lučjo je potrebno stanje z rumeno lučjo.
– Pri prehodu iz rdeče na zeleno luč je prisotno stanje, v katerem sta istočasno prižgani rdeča in rumena luč.

Relizacija logičnega avtomata v QCADesigner:

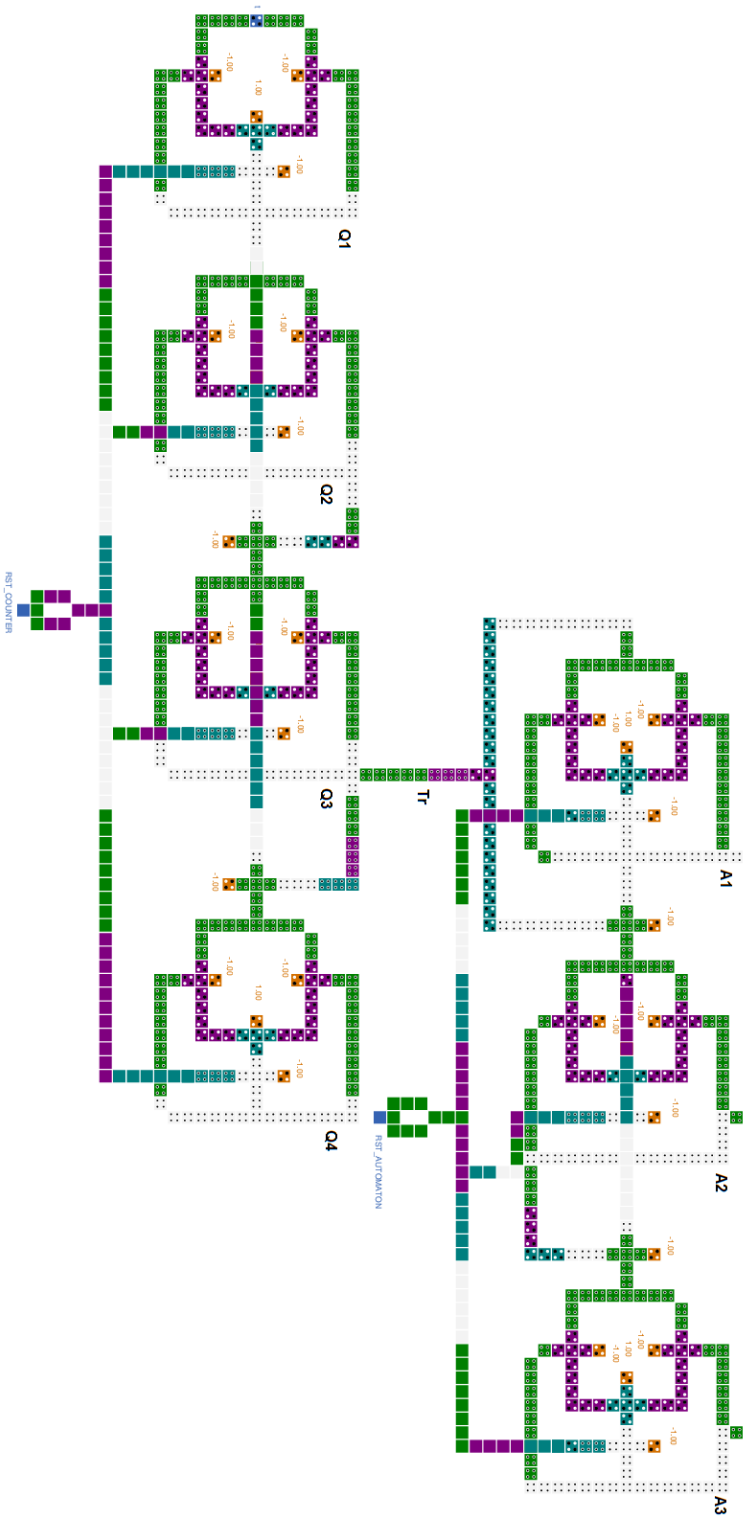


Slika 13. 3 bitni avtomat v QCADesigner-ju

Za realizacijo avtomata v QCADesigner-ju smo uporabili 452 celic.

Ko oba dela združimo, dobimo avtomat, ki deluje po zgoraj navedeni tabeli prehanja stanj in z upoštevanjem določenega zamika štirih urinih period.

Celoten avtomat v QCADesigner:



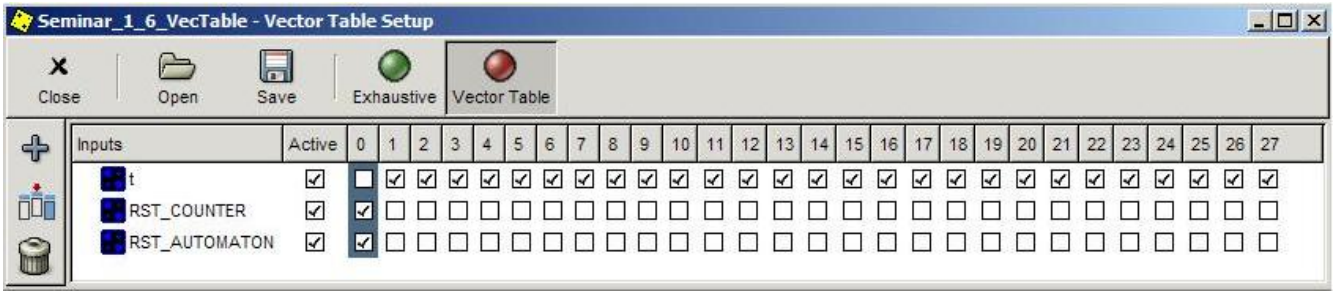
Slika 14. Avtomat ki je povezan s števcem

Rezultati

Kot je razvidno iz slike, priložene na naslednji strani, avtomat deluje kot predvideno v odstavku o realizaciji.

Z realizacijo avtomata smo zadovoljni. Ne samo zaradi pravilnega delovanja, ampak tudi zaradi po našem mnenju dobrega načrtovanja, ki omogoča spreminjanje trajanja zelene / rdeče luči na enostaven način.

Delovanje semaforja smo simulirali z testnim vektorjem:



Slika 15. Testni vektor za simulacijo avtomata

Opis delovanja

Semafor deluje po realnem modelu. Upošteva vse štiri faze semaforjev, ki jih srečamo na cestah:

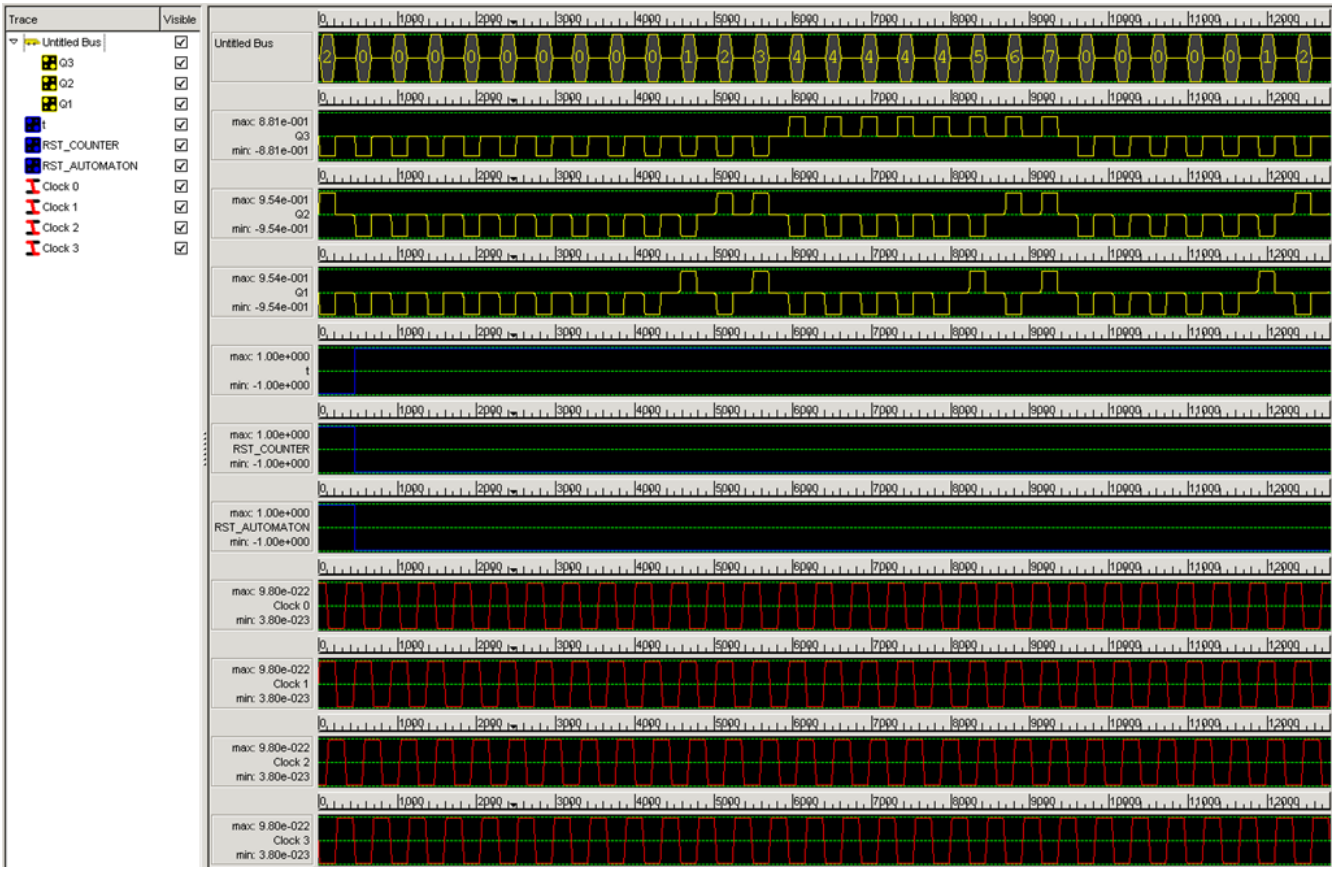
- Rdeče / zeleno stanje traja pet urinih period (eno interno stanje).
- Rdeče / rumeno stanje traja eno urino periodo (eno interno stanje).
- Rdeče / rdeče stanje traja eno urino periodo (eno interno stanje).
- Rdeče–rumeno / rdeče stanje traja eno urino periodo (eno interno stanje).

Kljub temu, da je število internih stanj ostalo isto, ima semafor več (zunanjih) stanj. To smo dosegli tako, da se vsako interno stanje preslika v različno zunanje stanje, (za razliko od prve implementacije, ko se npr. prva tri interna stanja preslikajo v isto zunanje stanje).

Tak način implementacije nam je omogočil števec, ki nam dovoljuje zadrževati stanje za poljubno število urinih period.

Perioda semaforja je šestnajst. Drugi semafor je od prvega zamaknjen za polovico periode, (prvo stanje na drugem semaforju je enako devetemu stanju na prvemu semaforju).

Rezultati simulacije:



Slika 16. Rezultati simulacije avtomata v QCADesigner-ju

Na sliki 16 je prikazana simulacija avtomata za kontrolo semaforja pri modularni implementaciji. Kot je razvidno iz slike 16, po začetni latenci # je avtomat v stanju 0 pet urinih period; v naslednji urini periodi se postavi v stanje 1 in ostane v tem stanje eno urino period; eno urino period trajata tudi stanji 2 in 3. V stanju 4 ostane pet urinih period. Stanja 5,6 in 7 trajajo eno urino periodo.
Iz stanja 7 preide v stanje 0 in cikel se ponovi.
Na sliki 16 je bolj podrobno prikazan isti potek dogodkov, kot na sliki 12.

IMPLEMENTACIJA PRESLIKOVALNE FUNKCIJE

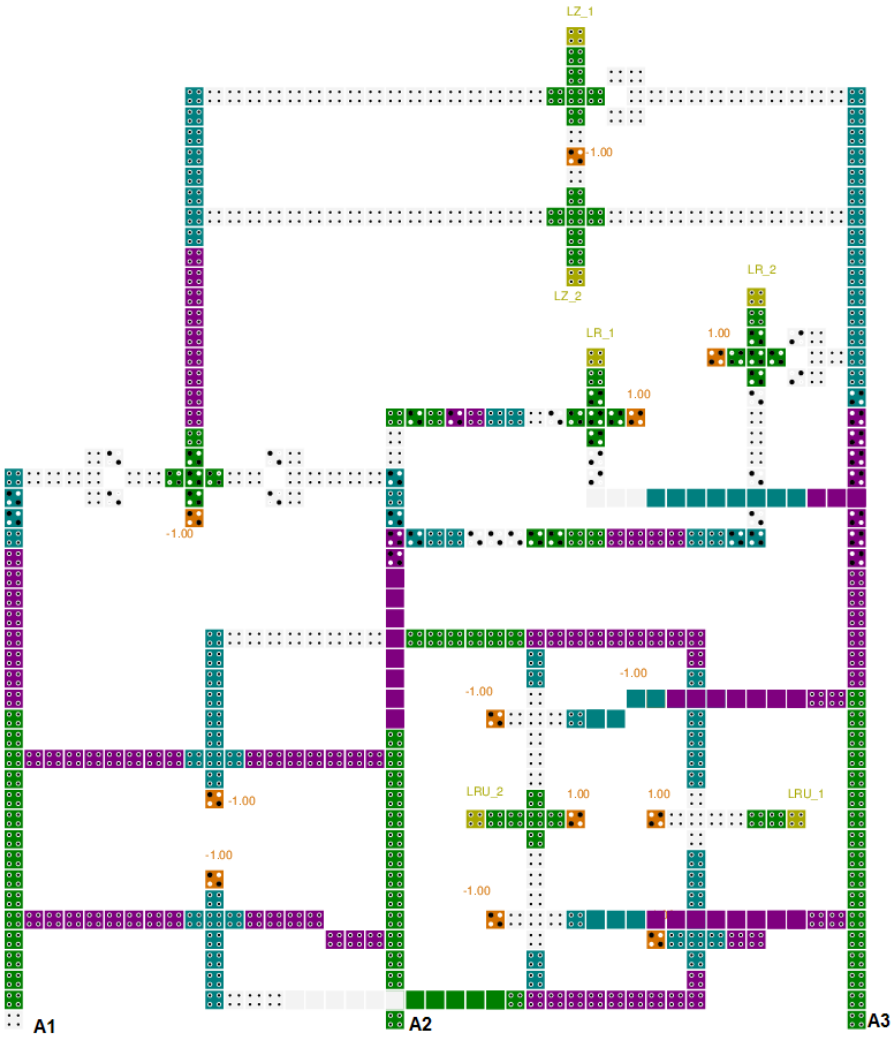
Po uspešno končanem avtomatu za krmiljenje semaforja smo realizirali še vezje, na katerega lahko neposredno priključimo luči semaforja.Vezje smo realizirali samo za modularno implementacijo.

Vezje relizira naslednje preklopne funkcije:

$LR_1 = A3 \vee A2$
$LRU_1 = (\neg A3 \wedge \neg A2 \wedge A1) \vee (A3 \wedge A2 \wedge A1)$
$LZ_1 = \neg A3 \wedge \neg A2 \wedge \neg A1$
$LR_2 = \neg A3 \vee A2$
$LRU_2 = (\neg A3 \wedge A2 \wedge A1) \vee (A3 \wedge \neg A2 \wedge A1)$
$LZ_2 = A3 \wedge \neg A2 \wedge \neg A1$

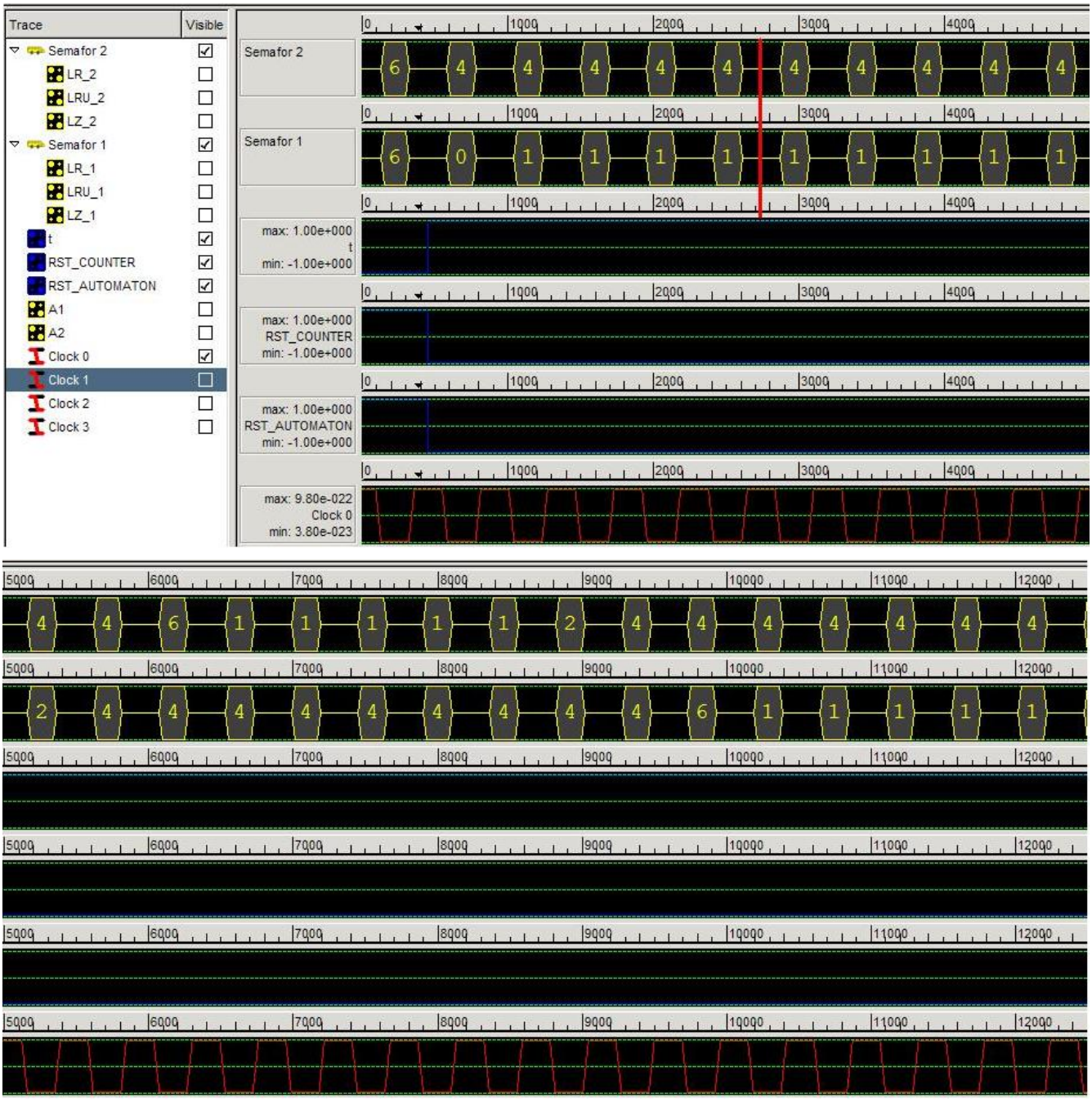
Podrobno razlago preklopnih funkcij si bralec lahko ogleda v podpoglavju “Izboljšana modularna implementacija” poglavja “Implementacija semaforja”.

Realizacija vezja v QCADesigner:



Slika 17. Vezje ki generira izhode

Rezultati simulacije:



Slika 18. Rezultati simulacije celotnega vezja v QCADesigner-ju

Na sliki 18 je prikazana simulacija celotnega vezja. Številka 1 (001) na izhodu pomeni, da na semaforju gori zelena luč, številka 4 (100) na izhodu pomeni, da gori rdeča luč, številka 2 (010) rumena luč, številka 6 (110) pa rdeča in rumena luč hkrati.

LASTNOSTI CELOTNEGA VEZJA (modularna implementacija)

Vezje je realizirano s 1489 celicami. Skupni časovni zamik vezja je 7 urinih period. Časovni zamik števca je 3 urine periode, časovni zamik avtomata pa je 2 urini periodi. Vezje, ki generira izhode ima časovni zamik 2 urini periodi. Pri realizaciji v QCADesigner-ju smo uporabili 3 nivoje.

ZAKLJUČEK

Pri načrtovanju avtomata smo se srečali s številnimi težavami, katere smo poskušali rešiti s pomočjo literature. Med samim delom smo načrt realizacije večkrat spremenili in popravili. Končni izdelek bi z nekoliko truda lahko še dodatno optimizirali, predvsem kar se tiče števila celic.

Avtomat bi lahko dodatno približali realnemu modelu, če bi implementirali še pomožni semafor za pešče in kolesarje, vendar pa to že presega obseg seminarske naloge.

Veliko težav pa nam je povzročal QCADesigner, predvsem z svojim nekonsistentnim delovanjem na različnih platformah. Po poenotenju razvojnih okolij (prehod na Windows 7) so bile težave odpravljene.

Med realizacijo same seminarske naloge smo svoje znanje o kvantnih celičnih avtomatih nadgradili z praktičnimi izkušnjami, kar je izboljšalo naše razumevanje celičnih avtomatov.

VIRI

[1] http://www.mina.ubc.ca/qcadesigner_manual

[2] Orač, T., 2007. Realizacija aritmetično-logičnih primitivov s strukturami kvantnih celičnih avtomatov.

[3] Kopina, A., Cerkvenik, M., Gorjanc, B., Laharnar, V., Mazalin, M., Pinterič, M., 2009. Avtomat za kontrolo križišča. seminarska naloga, Ljubljana.

[4] Mraz, M., 2010. Zapiski s predavanj: 2. Kvantni celični avtomati