UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET
KATEDRA ZA ELEKTRONIKU

PROGRAMABILNI INTERVAL TAJMER INTEL 8253

STUDENTI:
ILIĆ DARKO 9637
MARINOVIĆ SRĐAN 9397
MILOŠ LAZIĆ 8783

Niš, Maj 2005.

SADRŽAJ:

1. BROJAČI I TAJMERI

Tajmer je česta periferna komponenta u mikroračunarskim sistemima. Koristi se za procenu trajanja vremenskih intervala. Kao perifererna jedinica tajmer generiše signale u određenim vremenskim intervalima, ili meri vreme između dva spoljašnja događaja.

Tajmer meri vreme tako što broji taktne impulse na njegovom ulazu, čija je perioda poznata. Imaju i registar u koji korisnik postavlja jedan bit na osnovu koga interni multiplekser tipa 2×1 određuje koji će se taktni signal koristiti. Pri ovome kada se koristiti spoljašnji *clock* čip se ponaša kao tajmer, a kada se koristiti interni *clock* čip se ponaša kao brojač. Naredno funkcionalno proširenje je detektovanje isteka definisanog vremenskog intervala. To se ostvaruje tako što se u unutrašnjem registru upiše željena vrednost koja se zatim u svakom taktu upoređuje u komparatoru sa stanjem tajmera

Brojač je opštiji oblik tajmera. Brojači su *sekvencijalne logičke mreže* čiji dijagram stanja predstavlja repetitivni ciklus. Broj različitih stanja u ciklusu predstavlja moduo ili osnovu brojanja. Kao memorijski elementi u brojačima se koriste flip-flopovi. Ako se svi flip-flopovi u brojaču taktuju zajedničkim taktnim impulsom, takvi brojači se nazivaju sinhroni. Ako taktni impuls nije zajednički za sve flip-flopove, brojač je asinhroni.

Umesto da broji taktne impulse, on broji impulse od nekog drugog ulaznog signala. Često se kombinuju brojači i tajmeri u cilju merenja brzine, npr. brojimo obrtaje točka u toku jedne sekunde ili minuta da bi smo odredili brzinu kretanja automobila.

Da bi smo koristili tajmer/brojač potrebno je da poznajemo njegovu unutrašnju strukturu.. Brojač ima dodatni izlaz koji se aktivira kada brojač dođe u krajnje stanje, nakon čega pređe u stanje ″0000″, koje se naziva i prekoračenje. Izlaz može da se veže na *interrupt* ulaz procesora pa se u okviru *interrupt* rutine broje prekidi.

Da bi smo proširili opseg brojanja vezujemo kaskadno dva ili više brojača. Tada se drugi brojač taktuje izlazom prvog brojača, koji označava prekoračenje.

Brojač može da sadrži i *preskaler* tj. kolo koje služi da podeli frekvenciju ulaznog taktnog signala. *Preskaler* može da ima fiksnu vrednost deljenja, ali je najčešće programabilan tako da korisnik može da odredi sa koliko će se podeliti frekvencija, a sve u cilju povećanja opsega Tajmera.

Tajmer može da se realizuje softverski tako što se napiše programska petlja koja se sastoji od instrukcija čije je trajanje poznato. U okviru petlje se napiše onoliko instrukcija koliko je potrebno da bi trajanje njihovog izvršavanja bilo jednako željenom vremenskom intervalu. Ovakav metod nije efikasan i u ovoj primeni dolazi do izražaja efikasnost korišćenja spoljašnjeg tajmera za merenje vremenskog intervala.

Jedna posebna vrsta tajmera je *Watchdog* tajmer. On radi tako što umesto da generiše signal nakon *X* taktnih impulsa, očekuje da se njemu pošalje impuls svakih *X* taktnih impulsa, a ako se to ne dogodi *Watchdog* tajmer generiše signal koji ukazuje na grešku.

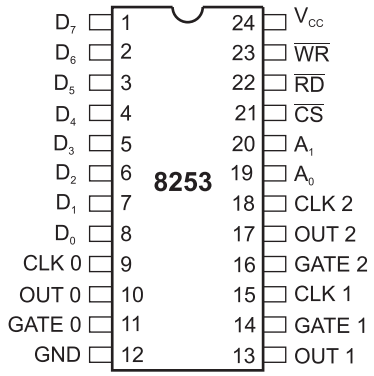
Ovaj tip tajmera se koristi u *embedded* sistemima i omogućava da se sistem restartuje u slučaju otkaza. Ako dođe do otkaza u programu usled ulaska u beskonačnu petlju ili čekanja na događaj koji neće nikad da se desi, *Watchdog* tajmer će odbrojati do kraja i generisati signal. Taj signal će resetovati ceo sistem ili izazvati prekid a u okviru prekidne rutine program se upućuje na neki bezbedan deo. Često se ove dve aktivnosti kombinuju tako što se u okviru prekidne rutine prvo izvršavanje programa uputi na neki bezbedan deo programa, testira se deo po deo sistema da bi se detektovala greška koja se memoriše, a zatim se resetuje ceo sistem.

2. PROGRAMABILNI INTERVAL TAJMER 8253

- **Kompatibilan sa 8-bitnom familijom mikroprocesora *INTEL 8085***
- **Poseduje tri 16-bitna brojača**
- **Maksimalna frekvencija taktne pobude do 2.6 MHz**
- **Programabilni režimi rada**
- **Brojanje binarno ili BCD**
- **Napajanje - jednostruko +5V**

INTEL 8253 je programabilni interval tajmer/brojač projektovan kao jedna od INTEL-ovih periferijskih komponenti. Izrađen je u NMOS tehnologiji sa jednostrukim +5V napajanjem u 24-pinskom plastičnom DIP kućištu.

Organizovan je kao tri nezavisna 16-bitna brojača, gde svaki ima maksimalnu frekvenciju taktne pobude od 2.6MHz. Svi modovi rada se softverski programiraju.



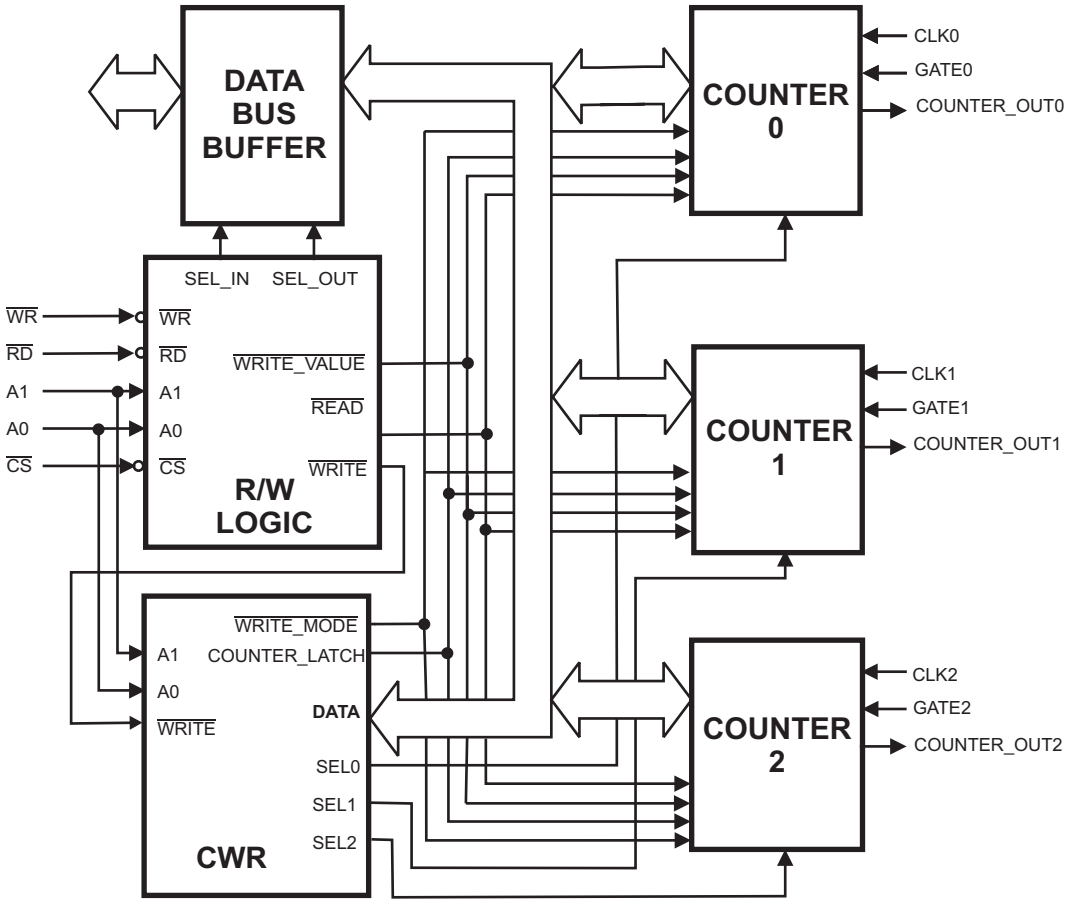
Slika 1. Raspored pinova na čipu 8253

2.1 OPIS KOLA

Programabilni interval tajmer/brojač 8253 je projektovan za primenu pre svega u *INTEL*-ovim mikroračunarskim sistemima. Uspešno rešava jedan od najčešćih problema koji se javljaju u svim mikroračunarskim sistemima, a to je generisanje preciznog vremenskog kašnjenja koje se može softverski kontrolisati. Umesto da formira vremenske petlje u sistemskom softveru, programer konfiguriše 8253 prema potrebi, inicijalizujući neki od njegovih brojača na željenu vrednost. Nakon komande za početak brojanja, 8253 će odbrojati željeni interval i poslati *interrupt* CPU-u. Time je smanjena obimnost programa i višestruka kašnjenja mogu biti lako organizovana dodeljivanjem prioriteta.

Pored generisanja određenog vremenskog kašnjenja, 8253 se može upotrebiti i za realizaciju nekih od sledećih sklopova :

- Programabilnog generatora brzine
- Brojača događaja
- Binarnog množača frekvencije
- Sata realnog vremena
- Kompleksnog kontrolera motora
- Delitelja frekvencije sa *N*



Slika 2. Blok šema 8253

BAFER PODATAKA

Trostatički, 8-bitni, bi-direkcioni bafer koristi se kao interfejs za povezivanje 8253 na sistemsku magistralu. Podaci se primaju i šalju preko bafera dok se izvršavaju ulazno-izlazne CPU operacije. Bafer se koristi prilikom upisivanja MOD-a, upisivanja inicijalnih vrednosti brojača i prilikom iščitavanja stanja brojača.

R/W LOGIC

Prihvata upravljačke signale sa sistemske magistrale, upravlja radom ulazno/izlaznog bafera i generiše signale za rad celokupnog kola. Rad bloka se kontroliše pomoću signala $\overline{CS}$.

$\overline{CS}$
Ulazni signal kojim se dozvoljava rad kola njegovim postavljanjem u stanje logičke nule. Ako kolo nije selektovano, neće se ni upisivati u njega niti čitati iz njega. Stanje ovog signala ne utiče na brojanje brojača.

$\overline{RD}$
Ulazni signal koji treba da bude aktivan (stanje logičke nule) dok CPU čita iz kola.

$\overline{WR}$
Ulazni signal koji treba da bude (stanje logičke nule) dok CPU upisuje u kolo.

A1,A0

Ulazi kojim se vrši selekcija jednog od tri brojača za upis inicijalne vrednosti za brojanje, ili *Control Word* registra za upis MOD-a brojanja.

Tabela 1. Tabela stanja za ulaze R/W logike

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	
0	1	0	0	0	učitavanje brojača 0
0	1	0	0	1	učitavanje brojača 1
0	1	0	1	0	učitavanje brojača 2
0	1	0	1	1	upisivanje *Control Word*-a
0	0	1	0	0	čitanje brojača 0
0	0	1	0	1	čitanje brojača 1
0	0	1	1	0	čitanje brojača 2
0	0	1	1	1	neaktivno stanje
1	x	x	x	x	neaktivno stanje
0	1	1	x	x	neaktivno stanje

***CONTROL WORD* registar**

CWR registar je selektovan kada su na ulazima stanja A0='1' i A1='1'. Tada se prihvataju informacije sa magistrale i smeštaju se u registar. Informacija smeštena u CWR sadrži selekciju odgovarajućeg brojača, MOD brojača i selekciju binarnog ili decimalnog brojanja. U njega je moguće samo upisivati sadržaj, bez mogućnosti njegovog iščitavanja.

COUNTER0,COUNTER1,COUNTER2

Ova tri brojača su potpuno identična i nezavisno se programiraju. Svaki se sastoji od 16-bitnog *DOWN* brojača sa mogućnošću učitavanja početne vrednosti i izbora decimalnog ili binarnog brojanja. Svaki može imati odvojen MOD za programiranje i vrstu brojanja, binarno ili BCD. Iščitavanje konteksta svakog brojača moguće je od strane programera korišćenjem jednostavne naredbe READ. Čak je omogućeno i "čitanje u letu" koje nema uticaj na brojanje brojača.

2.2 OPIS RADA

Kompletno funkcionisanje 8253 definisano je od strane softvera. Da bi se bilo koji brojač pustio u rad potrebno je da se upiše njegov MOD i inicijalna vrednost brojanja. Aktuelne operacije brojača su krajnje nezavisne što otklanja potrebu za dodatnim nadgledanjem rada kola.

Svaki brojač se individualno programira upisivanjem *Control Word*-a u CWR.

Tabela 2. Format Control Word-a

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BIN/BCD

Gde je značenje pojedinih bitova opisano tabelama koje slede:

Tabela 3. Selekcija brojača:

SC1	SC0	
0	0	selekcija brojača 0
0	1	selekcija brojača 1
1	0	selekcija brojača 2
1	1	nedozvoljeno stanje

Tabela 4. Upis/čitanje:

RW1	RW0	
0	0	lečovanje stanja brojača
0	1	čitanje/upisivanje brojača 0
1	0	čitanje/upisivanje brojača 1
1	1	čitanje/upisivanje brojača 2

Tabela 5. Izbor MOD-a

M2	M1	M0	
0	0	0	MOD0
0	0	1	MOD1
X	1	0	MOD2
X	1	1	MOD3
1	0	0	MOD4
1	0	1	MOD5

Tabela 6. Binarno/BCD brojanje:

BCD	
0	16-bitni binarni brojač
1	binarno kodovani decimalni (BCD) (4 dekade)

2.2.1 DEFINICIJA MOD-a RADA

MOD 0: Generisanje Interrupt-a po završetku brojanja

Izlaz brojača je inicijalno u stanju logičke nule, nakon upisivanja MOD-a. Upisivanje inicijalne vrednosti u brojač vrši se na prvu sledeću opadajuću ivicu CLK-a i počinje brojanje. Izlaz ostaje na logičkoj nuli dok brojač ne odbroji do kraja, a onda prelazi na logičku jedinicu i u tom stanju ostaje do upisivanja novog MOD-a ili nove inicijalne vrednosti. Tada se stanje brojača promeni na "FFFF" odnosno "9999" i nastavlja se brojanje.

Upisivanje neke vrednosti u registre brojača u toku brojanja izaziva:

1. upisivanje prvog bajta zaustavlja brojanje
2. upisivanje drugog bajta izaziva učitavanje nove vrednosti u brojač i početak novog ciklusa brojanja

Ulazni signal *GATE* zabranjuje brojanje kada je na logičkoj nuli odnosno dozvoljava brojanje kada je na logičkoj jedinici.

MOD1: Monostabilni multivibrator koji se hardverski ponovo okida

Izlaz brojača se postavlja inicijalno u stanje logičke nule nakon rastuće ivice ulaznog signala *GATE*. Kad brojač stigne do kraja, izlaz se postavlja u stanje logičke jedinice i tako ostaje do upisivanja novog MOD-a ili nove inicijalne vrednosti, a brojač nastavlja da broji. Upisivanje nove vrednosti u registre brojača nema uticaja na trenutno brojanje ukoliko se ne javi rastuća ivica signala *GATE.*

MOD2: Delitelj sa *N*

U ovom MOD-u brojač radi kao delitelj sa N. Izlaz brojača je inicijalno u stanju logičke jedinice i kada stigne do kraja prelazi u stanje logičke nule u trajanju od jedne periode CLK-a, i tada brojač automatski učitava vrednost iz ulaznih registara. Tako upisivanje nove vrednosti u ulazne registre nema uticaja na trenutni ciklus brojanja, ali će po završetku brojač učitati novu upisanu vrednost. Kada ulazni signal *GATE* pređe u stanje logičke jedinice, izlaz se odmah postavlja u stanje logičke jedinice. Brojač ne broji dok god je *GATE* na nuli, a njegova rastuća ivica izaziva učitavanje brojača i početak novog ciklusa brojanja.

MOD3: Generator pravougaonih impulsa

Sličan kao MOD2 s'tim što izlaz zadržava stanje logičke jedinice do isteka polovine ciklusa brojanja, a zatim prelazi u stanje logičke nule do kraja ciklusa brojanja. Ukoliko je u registre upisan neparan broj, stanje logičke jedinice će trajati jednu periodu CLK-a duže od stanja logičke nule, odnosno za upisan broj N, stanje logičke jedinice je (N+1)/2 dok logičke nule (N-1)/2.
Kada brojač odbroji do kraja, automatski se ponovo učitava. Ulazni signal *GATE* zabranjuje brojanje kada je na logičkoj nuli, inicijalizira brojanje rastućom ivicom i dozvoljava brojanje kada je na logičkoj jedinici.

MOD4: Softversko-trigerovani režim rada

Nakon upisivanja MOD-a, izlaz je u stanju logičke jedinice i kada brojač stigne do kraja izlaz prelazi u stanje logičke nule u dužini trajanja jedne periode CLK-a. Nakon upisivanja inicijalne vrednosti u ulazne registre, na prvu sledeću opadajuću ivicu CLK-a ta vrednost se učitava u brojač i otpočinje brojanje. Novi ciklus brojanja se započinje upisivanjem novog MOD-a ili nove inicijalne vrednosti.
Ulazni signal *GATE* zabranjuje brojanje kada je na logičkoj nuli odnosno dozvoljava brojanje kada je na logičkoj jedinici.

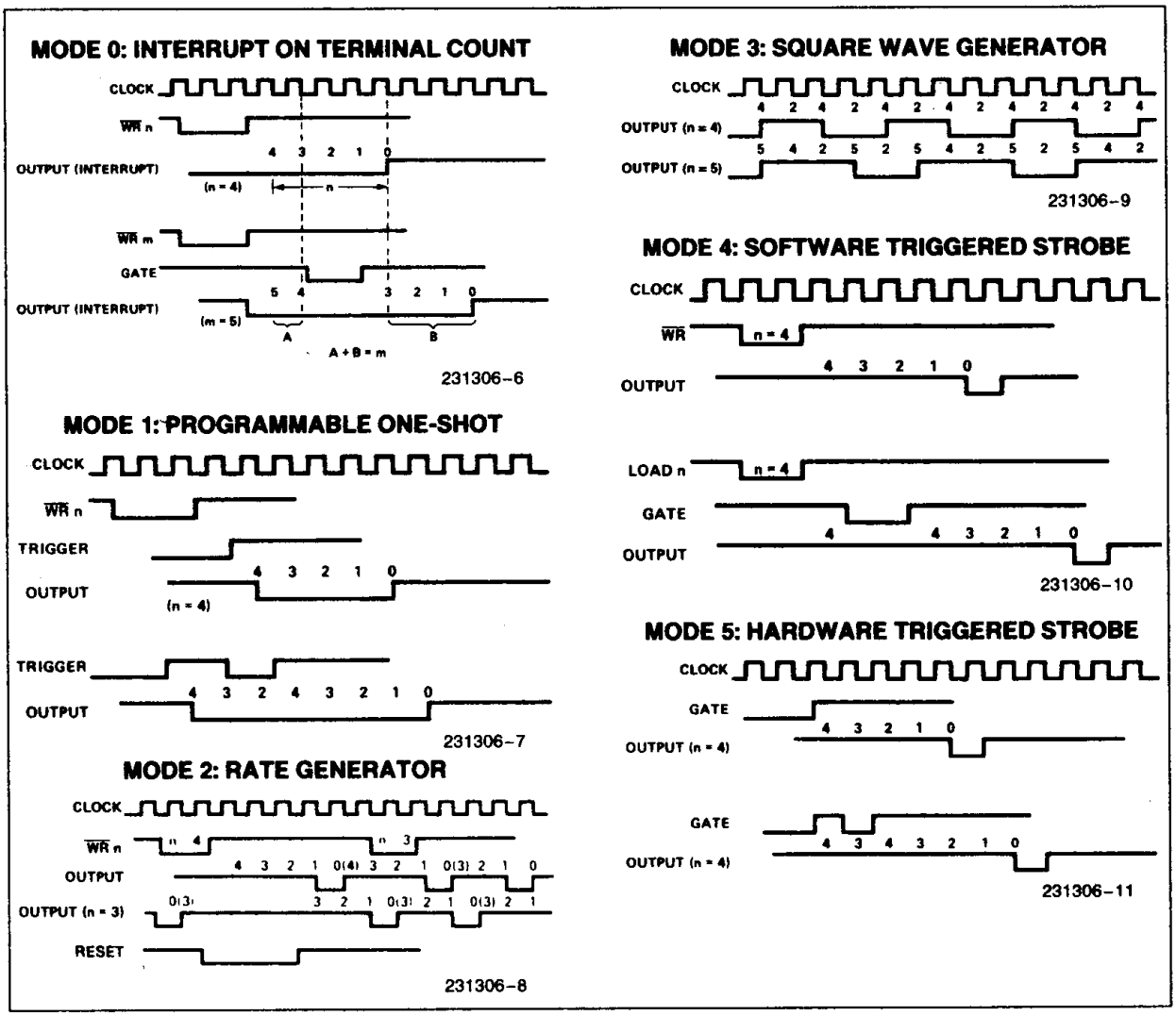
MOD5: Hardversko trigerovani monostabilni multivibrator

Nakon upisivanja MOD-a i inicijalne vrednosti, vrednost će biti učitana u brojač i počeće brojanje nakon rastuće ivice ulaznog signala *GATE*. Izlaz brojača je inicijalno u stanju logičke jedinice i kada brojač stigne do kraja, izlaz prelazi u stanje logičke nule jednu periodu CLK-a. Novi ciklus brojanja se inicijalizira upisivanjem novog MOD-a ili nove inicijalne vrednosti.

Sledeća tabela nam prikazuje uticaj stanja signala na ulazu *GATE* zavisno od stanja u kome se brojač trenutno nalazi:

Tabela 7

Stanje signala *GATE* Mod	logička nula ili opadajuća ivica	rastuća ivica	logička jedinica
0	zabranjuje brojanje	-	dozvoljava brojanje
1	-	1) inicira brojanje 2) resetuje izlaz nakon sledećeg CLK-a	-
2	1) zabranjuje brojanje 2) odmah postavlja izlaz na logičko 1	1) učitava brojač 2) inicijalizira brojanje	dozvoljava brojanje
3	1) zabranjuje brojanje 2) odmah postavlja izlaz na logičko 1	1) učitava brojač 2) inicijalizira brojanje	dozvoljava brojanje
4	zabranjuje brojanje	-	dozvoljava brojanje
5	-	inicijalizira brojanje	-



Slika 3. Vremenski dijagrami signala

2.2.2 PROCEDURA UPISIVANJA

Da bi neki od brojača mogao da se koristi potrebno je prvo upisati MOD rada upisivanjem odgovarajućeg *Control Word*-a kao i inicijalne vrednosti brojača.

Redosled programiranja je krajnje fleksibilan. Upisivanje MOD-a u *CONTROL WORD* registar može biti pri bilo kom selektovanom brojaču, znači ne mora se poštovati redosled upisivanja npr. brojač 0 pa brojač 1 pa brojač 2. Sadržajem *Control Word*-a određeno je na koji se brojač odnosi upisani MOD.

Upisivanje inicijalne vrednosti u brojač se mora izvršiti prema redosledu sadržanom u *Control Word*-u, bitno je da li se upisuje samo jedan bajt ili oba, i koji.

Ako je brojač programiran za upis dva bajta, prvo se upisuje LSB, pa onda MSB. Nakon upisivanja LSB-a sledeći podatak koji CPU upisuje u 8253 ne mora da bude MSB selektovanog brojača, ali on neće da počne da broji dok se ne završi procedura upisivanja odnosno dok se ne upiše i njegov MSB.

Pošto su brojači *DOWN* tipa, ako je upisan podatak "0000", posle prve opadajuće ivice CLK-a stanje će se promeniti u "FFFF" ako brojač broji binarno ili "9999" ako broji u BCD modu.

2.2.3 PROCEDURA ČITANJA

U brojnim aplikacijama javlja se potreba za iščitavanjem stanja brojača kako bi se dalje sprovela odgovarajuća izračunavanja. Opisano kolo sadrži dodatnu logiku koja omogućava programeru da lako iščita vrednosti svih brojača bez uticaja na sam proces brojanja.

Stanje brojača se može očitati na dva načina, a da se pri tome ne izazove greška u brojanju.

Prvi način čitanja je jednostavno selektovanje brojača čiji sadržaj želimo da iščitamo pomoću signala A0,A1 i dovođenje signala $\overline{\boldsymbol{RD}}$. Tada brojač zaustavlja brojanje do završetka procesa iščitavanja. Pri čemu se najpre vrši čitanje bajta manje težine LSB, a potom bajta veće težine MSB.

Drugi način je zadavanje komande za lečovanje stanja brojača, koje se vrši upisivanjem *Control Word*-a čiji su bitovi RW1='0' i RW0='0'. Tada izlazni registri zadržavaju trenutno stanje brojača, a brojač se ne zaustavlja. Nakon završetka procesa iščitavanja izlazni registri nastavljaju da prate stanje brojača.

Na taj način je izbegnuto da se stanje izlaznih registara menja u toku čitanja. Prilikom čitanja prvo se čita LSB pa MSB brojača.

Tabela 8. Selekcija brojača prilikom čitanja

A1	**A0**	**RD**	
0	0	0	čitanje brojača 0
0	1	0	čitanje brojača 1
1	0	0	čitanje brojača 2
1	1	0	nedozvoljeno stanje

Tabela 9. Izgled Control Word-a pri lečovanju

A0,A1 = 11

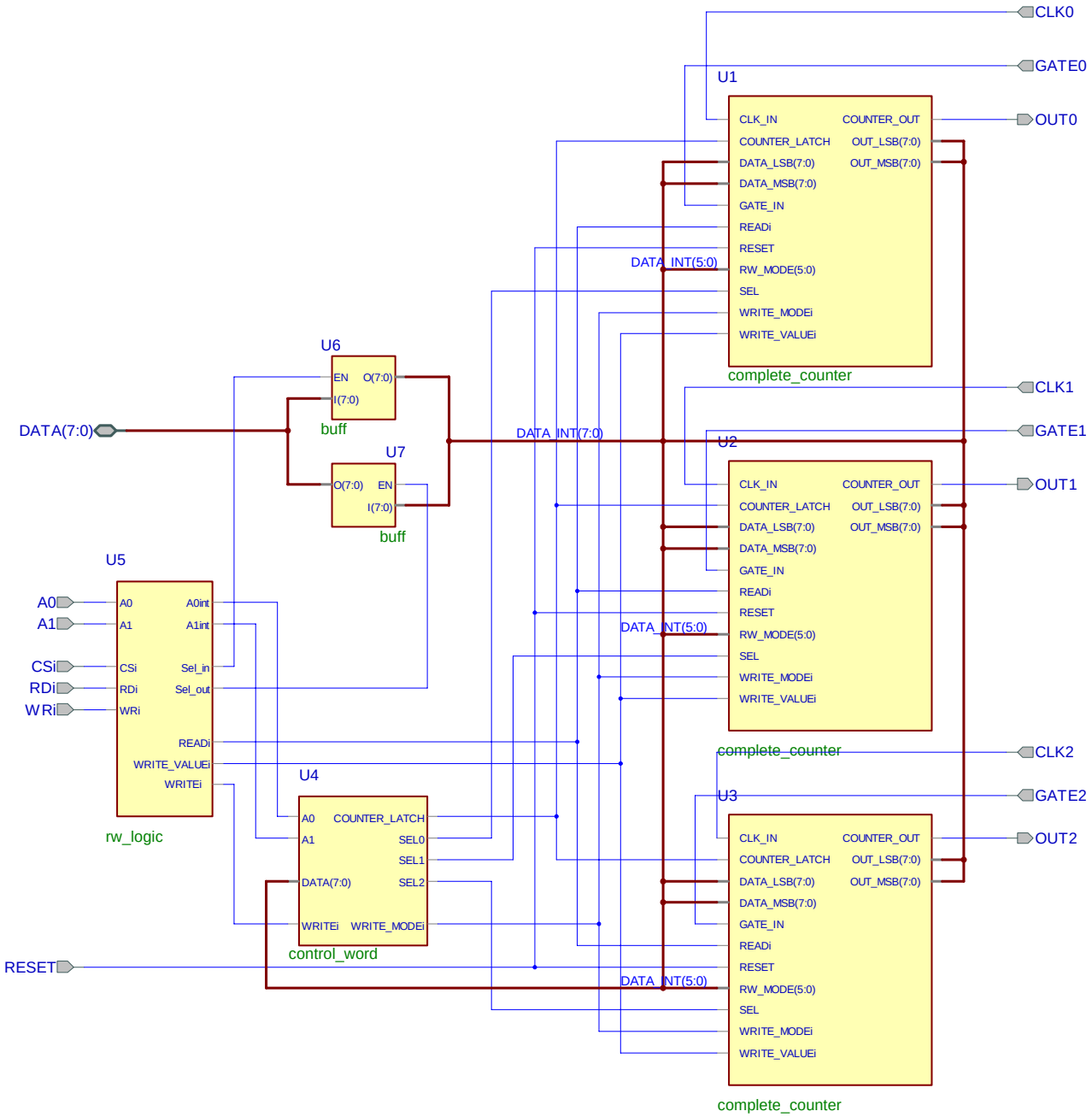
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

3. PREDLOG REALIZACIJE 8253 U VHDL-u

Prethodno pomenuto kolo je opisano u programskom jeziku VHDL koji je danas jedan od standardnih softverskih alata za projektovanje hardvera. Nakon toga je izvršena sinteza i implementacija opisa u programskom paketu *ISE* firme *XILINX*. Za sam razvoj i testiranje koda korišćen je programski paket *ACTIV-HDL* .

U strukturi na najvišem nivou razlikujemo 4 entiteta, kao na slici 3:

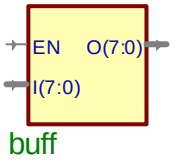
- entitet "*Complete_Counter*"
- entitet "*Control_Word*"
- entitet "*RW_logic*"
- entitet *"buff"*



Slika 4. BDE dijagram programabilnog tajmera/brojača 8253

3.1 **ULAZNO/IZLAZNI bafer** prenosi na svom izlazu signal koji mu je doveden na ulaz samo ako je signal dozvole (EN) aktivan tj. u stanju logičke jedinice. U suprotnom njegov izlaz je u stanju visoke impedanse. Ova mogućnost obezbedjuje njegovo povezivanje na zajedničku magistralu.
Listing kôd-a:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity buff is
      port (
            EN : in std_logic;
            I : in std_logic_vector(7 downto 0);
            O : out std_logic_vector(7 downto 0)
            );
end entity;

architecture buff_arch of buff is
begin

      O <= I when EN = '1' else (others => 'Z');

end buff_arch;
```



Slika 5. Blok "buff"

3.2 **R/W LOGIC** upravlja radom ulazno/izlaznog bafera tako što aktivira ulazni bafer kada su aktivni signali $\overline{WR}$ i $\overline{CS}$ (na logičkoj nuli), odnosno izlazni bafer kada su aktivni signali $\overline{RD}$ i $\overline{CS}$. Pored toga, generiše signal *WRITE_VALUE* kada se vrši upisivanje vrednosti u neki od brojača, i prenosi signale *READ* i *WRITE* kada je aktivan signal $\overline{CS}$.

Listing kôd-a:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity RW_logic is
      port(CSi: in std_logic;
           RDi: in std_logic;
           WRi: in std_logic;
           A0: in std_logic;
           WRITEi: out std_logic;
           A1: in std_logic;
           WRITE_VALUEi: out std_logic;
           READi: out std_logic;
           A1int: out std_logic;
           A0int: out std_logic;
           Sel_in: out std_logic;
           Sel_out: out std_logic);
end RW_logic;

architecture RW_logic of RW_logic is
component AND_2 is
      port(I0: in std_logic;
           I1: in std_logic;
           OUTPUT: out std_logic);
end component;
component OR_2 is
      port (
            I0 : in std_logic;
            I1 : in std_logic;
            O : out std_logic
      );
end component;
component INV_1 is
```

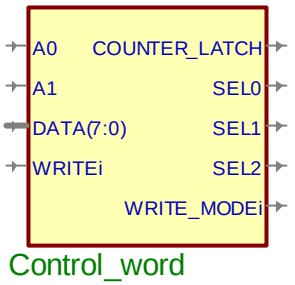
Slika6. Blok RW_logic

```
      port(q: in std_logic;
           notq: out std_logic);
end component;

signal NET1: std_logic;
signal NET2: std_logic;

begin
U1: AND_2
port map( I0=>A0,
          I1=>A1,
              OUTPUT=>NET1
              );
U2: OR_2
port map( I0=>WRi,
            I1=>NET1,
              O=>NET2
              );
U3: OR_2
port map( I0=>CSi,
          I1=>NET2,
              O=>WRITE_VALUEi
              );

    A1int<=A1 ;
      A0int<=A0 ;
      READi <= RDi when CSi='0' else '1';
      WRITEi <=WRi when CSi='0' else '1';
      Sel_in <='1'  when (CSi='0' and WRi='0') else '0';
      Sel_out <='1' when (CSi='0' and RDi='0') else '0';
end RW_logic;
```

3.3 **CONTROLWORD** je blok koji prihvata upravljačku reč *Control Word* i vrši dekodiranje sadržaja sedmog i šestog bita. Na osnovu njihove kombinacije vrši se izbor odgovarajućeg brojača. Brojač prihvata ostale bitove *Control Word*-a sa interne magistrale i smešta ih u svoj prihvatni registar. Na osnovu sadržaja ove informacije postavlja se režim rada MOD. Režim rada definiše da li će brojač brojati binarno ili BCD, da li će prihvatati upisivanje/čitanje višeg, nižeg ili oba bajta koja se odnose na njegov sadržaj. Ovaj blok generiše i signal za lečovanje stanja brojača prilikom operacije čitanja njegovog sadržaja. Operacija čitanja se može izvesti sa lečovanjem i bez lečovanja.

Listing kôd-a:

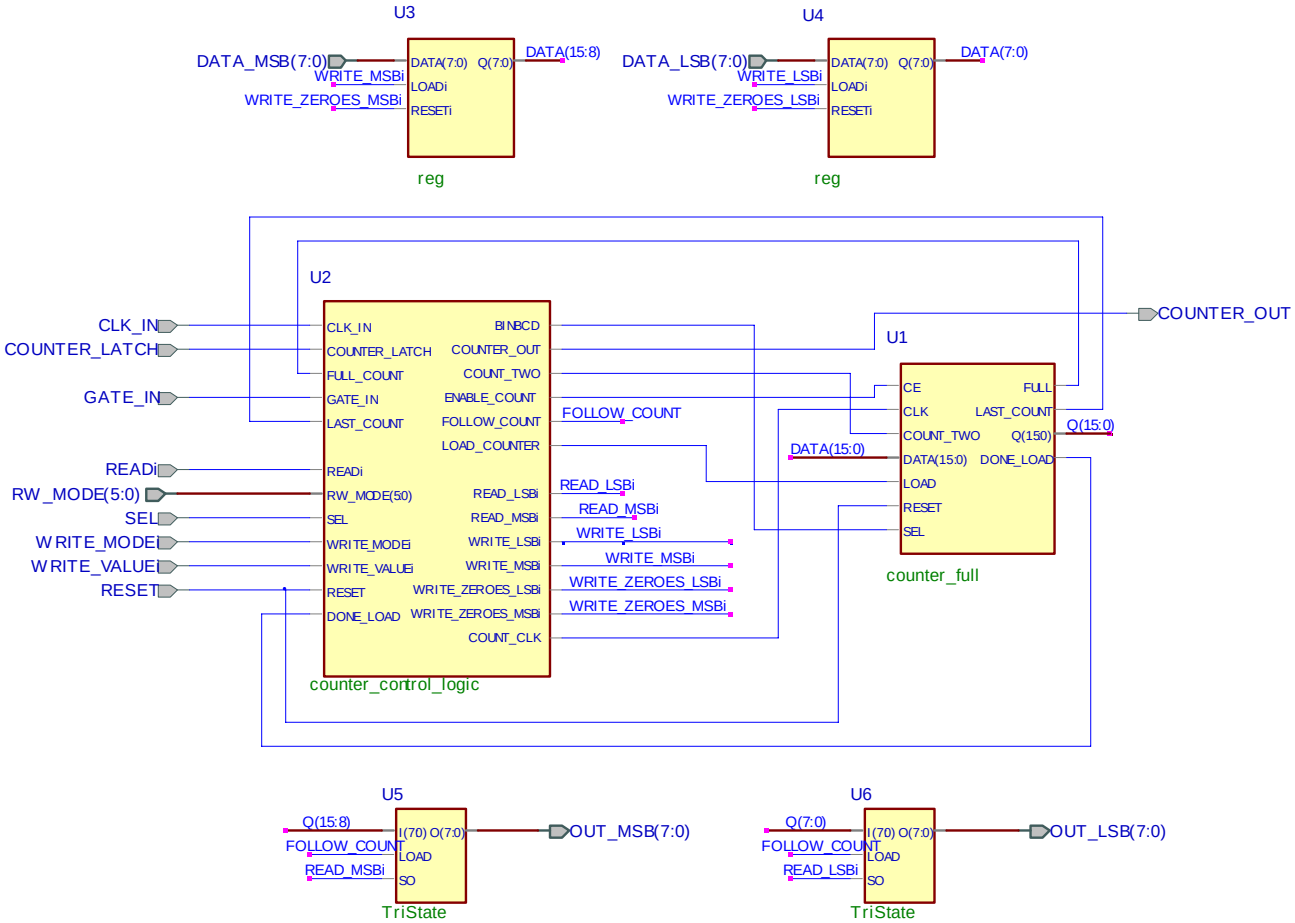
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Control_word is
      port (
            A1: in std_logic;
            A0: in std_logic;
            DATA: in std_logic_vector (7 downto 0);
            WRITEi: in std_logic;
            COUNTER_LATCH: out std_logic;
            WRITE_MODEi: out std_logic;
            SEL0: out std_logic;
            SEL1: out std_logic;
            SEL2: out std_logic
            );
end Control_word;

architecture Control_word of Control_word is

signal CONTROL_WORD:std_logic_vector (3 downto 0);
signal S0,S1,S2: std_logic;
```



Slika 7. Blok Control_Word

```
begin
      process(A1,A0,WRITEi,DATA,CONTROL_WORD)
begin
            if WRITEi='0' then
                  if A1='1' and A0='1'then
                        CONTROL_WORD <= DATA(7 downto 4);
                        WRITE_MODEi <= '0';
                        S0 <= (not CONTROL_WORD(3)) and (not CONTROL_WORD(2));
                        S1 <= (not CONTROL_WORD(3)) and CONTROL_WORD(2);
                        S2 <= CONTROL_WORD(3) and (not CONTROL_WORD(2));

                  end if;
            else
                  S0<='0';
                  S1<='0';
                  S2<='0';
                  WRITE_MODEi <='1';
            end if;
end process;
COUNTER_LATCH  <=  (not  CONTROL_WORD(1))  and  (not  CONTROL_WORD(0))  and  (not
WRITEi) ;

SEL0<='1' when S0='1' or (A1='0' and A0='0')  -- Selekcija nultog brojaca
else '0';
SEL1<='1' when S1='1' or (A1='0' and A0='1')  -- Selekcija prvog brojaca
else '0';
SEL2<='1' when S2='1' or (A1='1' and A0='0')  -- Selekcija drugog brojaca
else '0';

end Control_word;
```

3.4 COMPLETE_COUNTER



Slika 8. Blok Complete_Counter

Blok **Compete_counter** predstavlja jezgro brojača. U strukturi kola instancirana su tri identična bloka **Compete_counter**.

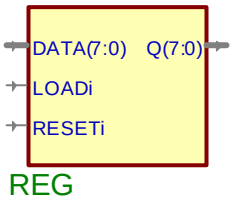
Kao što se vidi sa slike 8. gradivni blok **Compete_counter** čine sledeće celine:

- dva osmobitna ulazna registra, U4 i U3, koji prihvataju LS i MS bajtove podataka sa sistemske magistrale, respektivno
- dva trostatička registra, U5 i U6, čiji sadržaj odgovara trenutnom stanju brojača. U toku operacije čitanja bloka **Compete_counter** postavljaju svoj sadržaj na internu magistralu. Inače su u stanju visoke impedanse
- 16-bitni *DOWN* brojač, U1, koji može da broji binarno ili dekadno. Na kraju brojanja generiše signal FULL
- kontroler, U2, koji kompletno upravlja radom brojača

3.5 ULAZNI REGISTAR, U3 (U4), prihvata osmobitni podatak dostupan na svom ulazu ako je signal za učitavanje *LOAD* aktivan (logička nula). Logičkom nulom signala *RESET* vrši se resetovanje ovog bloka.

Listing kôd-a:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity REG is
      port (
            LOADi : in std_logic;
            RESETi: in std_logic;
            DATA : in std_logic_vector(7 downto 0);
            Q: out std_logic_vector(7 downto 0));
end entity;

architecture REG_arch of REG is

      signal TEMP_Q: std_logic_vector(7 downto 0):="00000000";

begin
      process(LOADi,DATA,RESETi)
      begin
            if RESETi='0' then
                  TEMP_Q <= (others => '0');
            elsif LOADi = '0' then
                  TEMP_Q <= DATA;
                  end if;
      end process;
      Q<=TEMP_Q(7 downto 0);
end REG_arch;
```

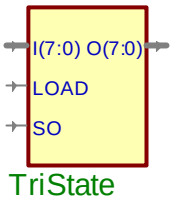


Slika 9. Blok REG

3.6 TROSTATIČKI REGISTAR, U5 (U6), prihvata osmobitnu informaciju sa svog ulaza kada je aktivan (logička jedinica) signal za učitavanje *LOAD*. Ako je signal za dozvolu rada izlaza *SO* aktivan (logička nula) tada se stanje ovog registra prosleđuje na njegovom izlazu. U suprotnom, izlaz registra je u stanju visoke impedanse.

Listing kôd-a:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity TriState is
      port (
            SO : in std_logic;
            LOAD: in std_logic;
            I : in std_logic_vector(7 downto 0);
            O : out std_logic_vector(7 downto 0)
            );
end entity;

architecture buff_arch of TriState is

signal TEMP: std_logic_vector(7 downto 0):="ZZZZZZZZ";

begin

process(LOAD,SO,I,TEMP)

begin

      if LOAD = '1' then
            TEMP <= I;
      end if;

      if SO='0' then
```

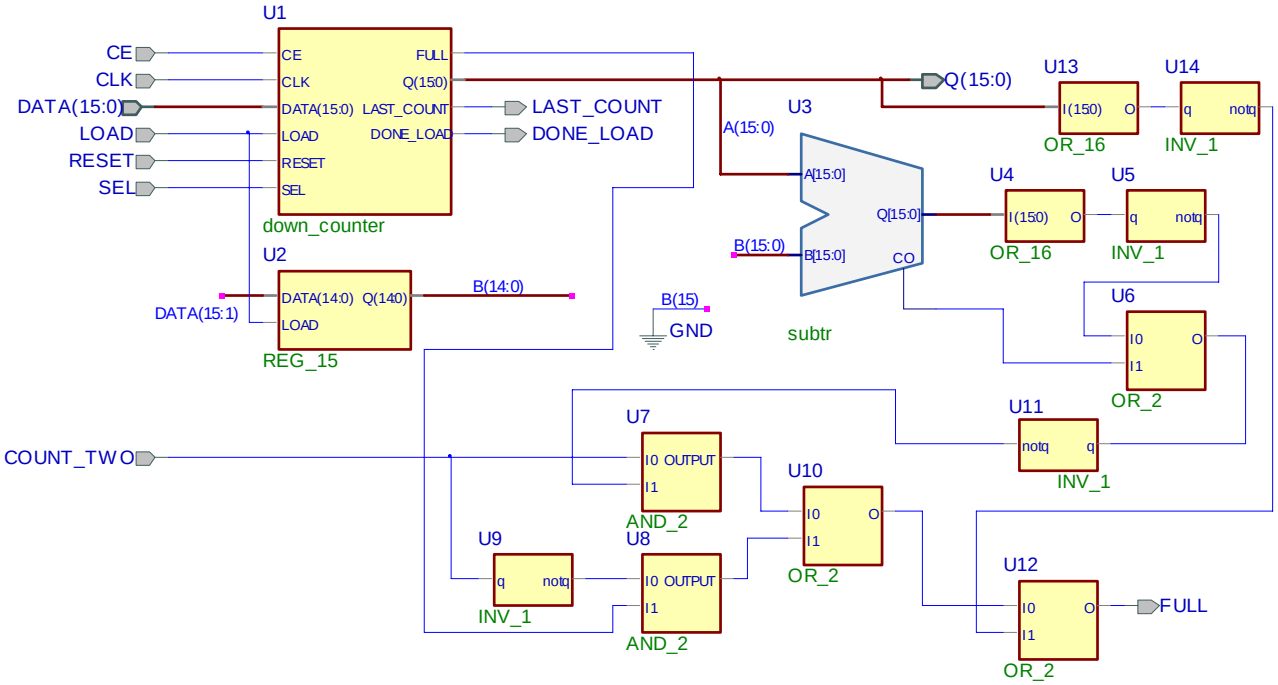


Slika 10. Blok "TriState"

```
            O <= TEMP;
        else
              O <=(others => 'Z');
        end if;
end process;
end buff_arch;
```

3.7 COUNTER FULL

Kao što se vidi na slici 11. gradivni blok **COUNTER FULL** čine brojač U1 i ostala logika, blokovi od U2 do U14, koja je potrebna za realizovanje MOD-a 3.



Slika 11. Blok Counter Full

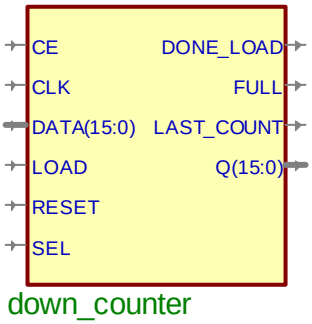
3.7.1 DOWN_COUNTER predstavlja 16-bitni kombinovani binarni/BCD brojač koji broji unazad. Dozvola brojanja se vrši preko ulaza CE. Selekcija načina brojanja, binarno ili BCD, definiše se preko ulaza SEL. Stanje logičke nule znači da se radi o binarnom, a stanje logičke jedinice da se radi o BCD brojanju. Ulaznim signalom LOAD vrši se postavljanje brojača. Nakon upisa aktivira se signal DONE_LOAD. Stanje brojača se menja na opadajućoj ivici CLK-a. Kada je stanje brojača jedan (stanje "0000 0000 0000 0001") generiše se signal LAST_COUNT.

U MOD-u 3 izlazni signal je u stanju logičke jedinice do polovine modula brojanja, a u stanju logičke nule od polovine do kraja. Ovaj zahtev je uslovio da se uvede dodatni kontrolni signal, COUNT_TWO i dodatna logika, U2 do U14.

Moduo brojanja se upisuje u 15-bitni prihvatni registar, U2, pri čemu se MS bit zanemaruje, što je ekvivalentno pomeranju u desno, odnosno deljenju sa dva. Sadržaj registra U2 se zatim dovodi na jedan od ulaza 16-bitnog oduzimača, U3, i oduzima od trenutne vrednosti brojača U1. Izlaz CO, bloka U3, postaje jedinica kada je razlika negativna, čime se detektuje polovina vrednosti brojanja. Signal CO se dovodi na dvoulazno OR kolo,U6. Drugi ulaz kola U6 je marker uslova *Zero* koji detektuje kada je izlaz kola U3 nula. Signal FULL je u stanju logičke jedinice u toku prve polovine modula brojanja, a nula u drugoj polovini. Signal COUNT_TWO odredjuje da li će se aktivirati logika za MOD3.

Listing kôd-a za blok "down_counter":

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity down_counter is
      port (
            CLK : in std_logic;
            RESET : in std_logic;
            CE : in std_logic;
            SEL : in std_logic;
            LOAD : in std_logic;
            DATA : in std_logic_vector(15 downto 0);
            Q : out std_logic_vector(15 downto 0);
            LAST_COUNT: out std_logic;
            DONE_LOAD: out std_logic:='1';
            FULL: out std_logic
      );
end entity;
```



Slika 12. Blok "Counter"

```
architecture down_counter_arch of down_counter is
signal STANJE_BROJACA : std_logic_vector(15 downto 0);
signal DECADE_TEMP_CE : std_logic_vector(3 downto 0);

begin
      process(CLK,RESET,CE,SEL,DATA)
      begin
      if CE='1' then           -- dozvola rada
            if RESET = '1' then      -- resetovanje brojaca
                  STANJE_BROJACA <= (others => '0');
            else
                  if CLK'event and CLK='0' then
                  if LOAD='1' then   --   ucitavanje stanja u brojac
                        STANJE_BROJACA <= DATA;
                        DONE_LOAD<='0';
                  elsif SEL='0' then  -- binarno brojanje
                         STANJE_BROJACA <= STANJE_BROJACA - 1;
                         DONE_LOAD<='1';
                    elsif SEL='1' then  -- BCD brojanje
                       DONE_LOAD<='1';
                        -- Prva cifra
                    if DECADE_TEMP_CE(0) = '1' then
                              if STANJE_BROJACA(3 downto 0) = "0000" then
                                    STANJE_BROJACA(3 downto 0) <= "1001";
                              else
                  STANJE_BROJACA(3 downto 0) <= STANJE_BROJACA(3 downto 0) - 1;
                              end if;
                        end if;
                        -- Druga cifra
                        if DECADE_TEMP_CE(1) = '1' then
                              if STANJE_BROJACA(7 downto 4) = "0000" then
                                    STANJE_BROJACA(7 downto 4) <= "1001";
                              else
                  STANJE_BROJACA(7 downto 4) <= STANJE_BROJACA(7 downto 4) - 1;
                              end if;
                        end if;
                        -- Treca cifra
                        if DECADE_TEMP_CE(2) = '1' then
                              if STANJE_BROJACA(11 downto 8) = "0000" then
                                    STANJE_BROJACA(11 downto 8) <= "1001";
                              else
                 STANJE_BROJACA(11 downto 8) <= STANJE_BROJACA(11 downto 8) - 1;
                              end if;
                        end if;
                        -- Cetvrta cifra
                   if DECADE_TEMP_CE(3) = '1' then
                            if STANJE_BROJACA(15 downto 12) = "0000" then
                                  STANJE_BROJACA(15 downto 12) <= "1001";
                            else
               STANJE_BROJACA(15 downto 12) <= STANJE_BROJACA(15 downto 12) - 1;
                            end if;
```

```
                    end if;
                end if;
        end if;
end if;
end if;
end process;
    DECADE_TEMP_CE(0) <= '1';
      DECADE_TEMP_CE(1) <= '1' when STANJE_BROJACA(3 downto 0) = "0000" else
'0';
      DECADE_TEMP_CE(2) <= '1' when STANJE_BROJACA(7 downto 4) = "0000" and
DECADE_TEMP_CE(1)='1' else '0';
      DECADE_TEMP_CE(3) <= '1' when STANJE_BROJACA(11 downto 8) = "0000" and
DECADE_TEMP_CE(2)='1' else '0';
      FULL <='1' when STANJE_BROJACA ="0000000000000000" else '0';   -- '1' kad
je brojac   u nuli
      LAST_COUNT <= '1' when STANJE_BROJACA ="0000000000000001" else '0';     --
'1' kad je brojac u jedinici
      Q <= STANJE_BROJACA;
      end architecture;
```

3.8 COUNTER_CONTROL_LOGIC je sekvencijalni logički blok koji na osnovu stanja signala na ulazu upravlja radom brojača generišući upravljačke signale potrebne za njegov rad.

Prilikom upisa novog MOD-a (aktivan signal WRITE_MODEi) prihvata se deo *Control Word*-a (bitovi 3,..,1). Na osnovu LS bita *Control Word*-a odredjuje se da li će brojač brojati binarno ili BCD (pomoćni signal BINBCD*)*, dok bit pozicije 5 i 4 *Control Word*-a određuju koji bajt mora da bude upisan u fazi inicijalizacije (pomoćni signal RW1RW0*)*.

Prilikom upisa nove vrednosti brojanja (aktivan signal WRITE_VALUEi) generišu se signali za upisivanje ulaznih registara (WRITE_LSBi ili WRITE_MSBi). Ako se upisuje vrednost za brojanje koja se sastoji od samo jednog bajta potrebno je resetovati drugi bajt, jer je u njemu sadržana vrednost koja je upisana u prethodnom ciklusu. Zato se generiše signal WRITE_ZEROES_MSBi prilikom upisa LSB-a, kada je brojač programiran za upis samo LSB-a, koji se vodi na ulaz za reset MSB registra. Kada je brojač programiran za upis samo MSB-a, tada se prilikom upisa MSB-a generiše signal WRITE_ZEROES_LSBi koji se vodi na ulaz za reset LSB registra.

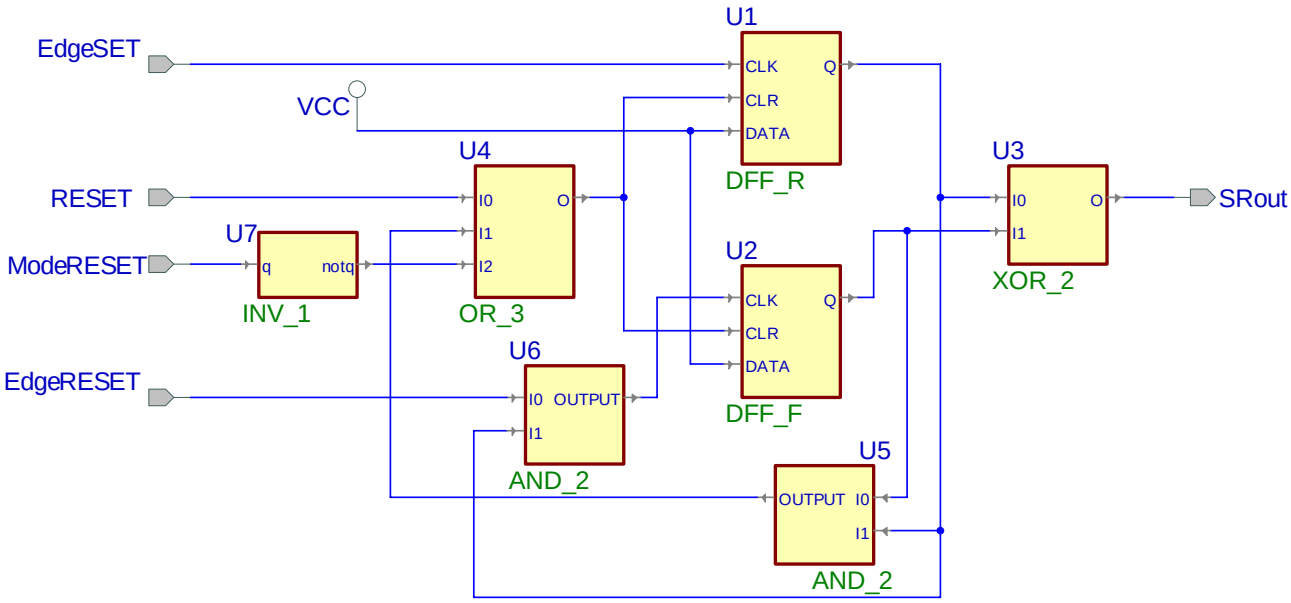
Prilikom upisa u ulazne registre generiše se i signal DONE_WRITE koji označava da je završen proces upisivanja. Prilikom upisivanja dva bajta signal DONE_WRITE postaje nula nakon što počne proces upisivanja prvog bajta a postaje jedinica kada započne upisivaje drugog bajta. Ovaj signal se kasnije koristi za zabranu brojanja brojača.

Prilikom čitanja, uvek se čitaju oba bajta i to prvo LSB pa zatim MSB, tako da kontroler generiše signale za dozvolu rada izlaznih trostatičkih registara i to za LSB pri prvom aktivnom stanju ulaznog signala RDi, a za MSB pri drugom aktivnom stanju ulaznog signala RDi.

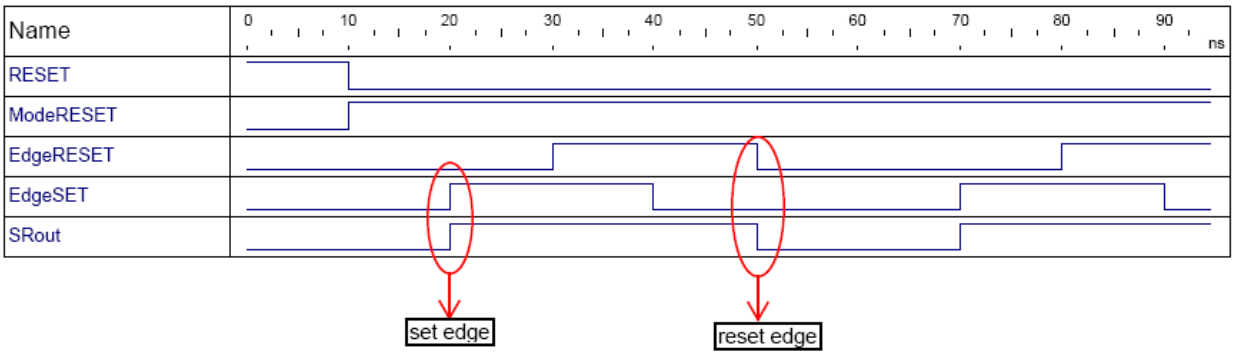
Za vreme čitanja se zabranjuje brojanje brojača ako prethodno nije inicirana komanda za lečovanje registara. Brojanje se zabranjuje kada se pojavi prvi ulazni signal RDi, a dozvoljava nakon očitavanja drugog bajta. Ako je inicirana komanda za lečovanje izlaznih registara, stanje registara se lečuje i oni ne prate više stanje brojača. Nakon toga, prilikom čitanja, brojač se ne zaustavlja a po očitavanju oba bajta, izlazni registri nastavljaju da prate stanje brojača.

Pri opisivanju ovog bloka javila se potreba za kolom koje će se direktno setovati rastućom ivicom signala EdgeSet, a direktno resetovati opadajućom ivicom signala EdgeReset. Pored toga može se direktno resetovati logičkom jedinicom preko ulaza RESET ili logičkom nulom preko ulaza MODE_RESET. Opis takvog kola nije bio moguć na nivou opisa ponašanja, pa je takvo kolo strukturno realizovano i njegova struktura je prikazana na slici 13. Na slici 14. su prikazani vremenski dijagrami ovog kola. Ovo kolo je najčešće instancirano na mestima gde je bilo potrebno jedan signal setovati spoljnim asinhronim signalom a resetovati ga unutrašnjim sinhronim signalom.

Napomena: U prethodnom tekstu skraćenice MSB i LSB se odnose na bajt veće i manje težine u okviru 16-bitne reči, respektivno.



Slika 13. Blok "DoubleEdgeSR"

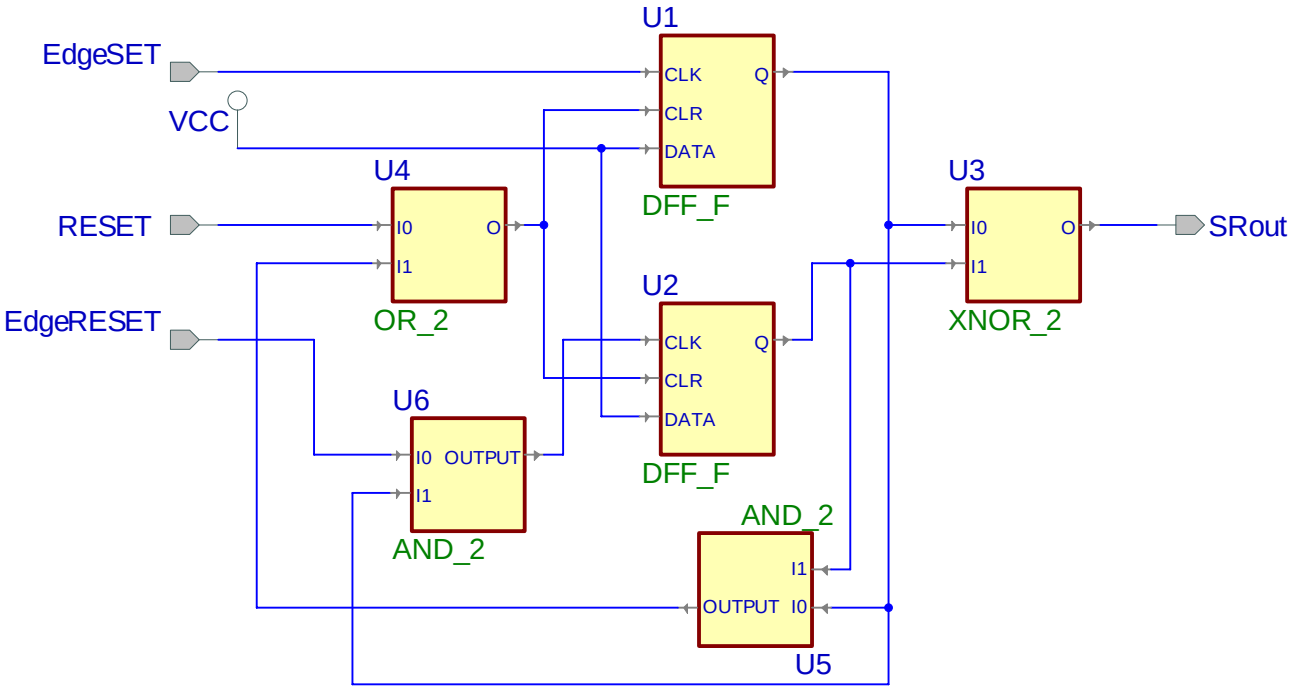


Slika 14. Vremenski dijagram DoubleEdgeSR-a

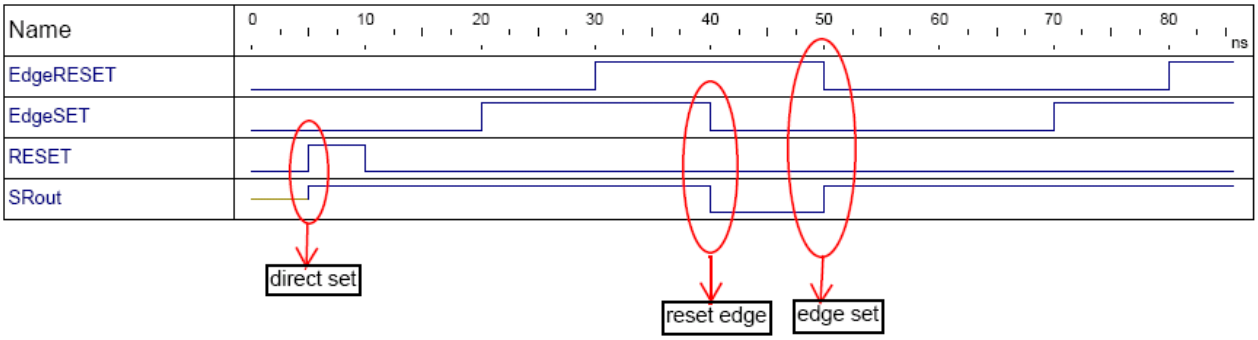
Sa slike 14. vidimo da se izlazni signal SRout postavlja u stanje logičke jedinice rastućom ivicom signala EdgeSET, a resetuje se opadajućom ivicom signala EdgeRESET. Resetovanje celokupnog kola vrši se pomoću signala RESET i ModeRESET, gde RESET treba da je na logičkoj jedinici, a ModeRESET na logičkoj nuli.

Definisano je i kolo koje će se direktno setovati opadajućom ivicom signala EdgeSET, a direktno resetovati opadajućom ivicom signala EdgeRESET i ima mogućnost spoljašnjeg setovanja logičkom jedinicom na ulazu RESET. Na slici 15. je prikazana struktura a na slici 16. prikazani su vremenski dijagram takvog kola.

Sa vremenskog dijagrama vidimo da je izlazni signal SRout nakon rastuće ivice signala RESET postavljen u stanje logičke jedinice. Pojavom opadajuće ivice signala Edge SET, SRout prelazi na logičku nulu i ostaje sve do pojave opadajuće ivice signala EdgeRESET.



Slika 15. Blok FallingEdgeSR



Slika 16. Vremenski dijagram FallingEdgeSR-a

Listing kôd-a:

```
library ieee;
use ieee.std_logic_1164.all;

entity COUNTER_CONTROL_LOGIC is
      port (
      CLK_IN: in std_logic;
      RESET: in std_logic;
      GATE_IN: in std_logic;
      FULL_COUNT:in std_logic;
      RW_MODE: in std_logic_vector(5 downto 0);
      WRITE_MODEi: in std_logic;
      SEL: in std_logic;
      READi: in std_logic;
      WRITE_VALUEi: in std_logic;
      COUNTER_LATCH: in std_logic;
      LAST_COUNT: in std_logic;
      DONE_LOAD: in std_logic;
      COUNT_TWO: out std_logic;
      WRITE_LSBi: out std_logic;
      WRITE_MSBi: out std_logic;
      WRITE_ZEROES_MSBi: out std_logic;
      WRITE_ZEROES_LSBi: out std_logic;
      READ_LSBi: out std_logic;
      READ_MSBi: out std_logic;
      FOLLOW_COUNT: out std_logic;
      COUNT_CLK: out std_logic;
```

```
      LOAD_COUNTER:out std_logic;
      ENABLE_COUNT:out std_logic;
      BINBCD: out std_logic;
      COUNTER_OUT: out std_logic
      );
end entity COUNTER_CONTROL_LOGIC;

architecture FSM of COUNTER_CONTROL_LOGIC is

component DoubleEdgeSR
      port(
       EdgeRESET : in STD_LOGIC;
       EdgeSET : in STD_LOGIC;
         ModeRESET : in STD_LOGIC;
       RESET : in STD_LOGIC;
       SRout : out STD_LOGIC
  );
end component;

component FallingEdgeSR
       port(
       EdgeRESET : in STD_LOGIC;
       EdgeSET : in STD_LOGIC;
       RESET : in STD_LOGIC;
       SRout : out STD_LOGIC
  );
end component ;
signal MODE:std_logic_vector(2 downto 0);--MOD brojaca
signal DONE_WRITE:std_logic:='1';-- zavrsen upis u kolo(1) ili ne(0)
signal  LOADED_COUNTER:std_logic:='1';--  zavrsen  upis  prilikom  upisivanja  dva
bajta
signal LOADED_LSB:std_logic:='0';--upisan LSB u CR (1)
signal ENABLE_FROM_READ:std_logic;--dozvola brojanja od strane citanja(1)
signal  COUNTER_LATCH_COMMAND:std_logic;--primljena  je  komanda  za  lecovanje  LR
reg.(0)
signal READED_COUNTER:std_logic;--ocitan je ceo brojac (LSB i MSB) (1)
signal READED_LSB:std_logic:='0';--pomocni siglal prilikom citanja oba bajta
signal LOADED_NEW: std_logic; --ucitana je nova vrednost u brojac(1) ili ne (0)
signal LOAD_NEW0: std_logic;  --signal za ucitavanje brojaca u modu 0
signal LOAD_NEW1: std_logic;  --signal za ucitavanje brojaca u modu 1
signal LOAD_NEW2: std_logic;  --signal za ucitavanje brojaca u modu 2
signal LOAD_NEW3: std_logic;  --signal za ucitavanje brojaca u modu 3
signal LOAD_NEW4: std_logic;  --signal za ucitavanje brojaca u modu 4
signal LOAD_NEW5: std_logic;  --signal za ucitavanje brojaca u modu 5
signal notCOUNTER_LATCH: std_logic;--invertovan signal COUNTER_LATCH
signal notREADED_COUNTER: std_logic;--    invertovan signal READED_COUNTER
signal SETOUT:std_logic;
signal SETOUT1:std_logic;           --LOAD_NEW2 posle ivice GATE_IN
signal SETOUT2:std_logic;           --LOAD_NEW2 posle zavrsetka brojanja
signal SETOUT3:std_logic;           --LOAD_NEW3 posle ivice GATE_IN
signal SETOUT4:std_logic;           --LOAD_NEW3 posle zavrsetka brojanja
signal notGATE_IN:std_logic;  --invertovan signal GATE_IN
signal RD_LSB:std_logic:='1'; --signal za citanje LSB
signal RD_MSB:std_logic:='1'; --signal za citanje MSB
signal notRD_LSB:std_logic;         --invertovan signal RD_LSB
signal notRD_MSB:std_logic;         --invertovan signal RD_MSB
signal RW1RW0:std_logic_vector(1 downto 0);--samo LSB,samo MSB ili oba
signal INICIAL_MODE:std_logic;      --signal '1' do inicijalizacije brojaca
signal START_LOAD:std_logic;  -- signal za ucitavanje brojaca
signal notSTART_LOAD:std_logic;     --inverovan signal za ucitavanje brojaca
signal notWRITE_MODE:std_logic;     --invertovan signal za upis moda
signal RESET0:std_logic:='0'; --signal za resetovanje signala LOAD_NEW0
signal RESET1:std_logic:='0'; --signal za resetovanje signala LOAD_NEW1
signal RESET2:std_logic:='0'; --signal za resetovanje signala LOAD_NEW2
signal RESET3:std_logic:='0'; --signal za resetovanje signala LOAD_NEW3
signal RESET4:std_logic:='0'; --signal za resetovanje signala LOAD_NEW4
signal RESET5:std_logic:='0'; --signal za resetovanje signala LOAD_NEW5
signal notRESET:std_logic;          --invertovan signal RESET
signal SETOUT11:std_logic;
begin

       U1 : DoubleEdgeSR                        --generisanje LOAD_NEW0
  port map(
```

```
      SRout => LOAD_NEW0,
      EdgeSET => DONE_WRITE,
        ModeRESET =>RESET0,
      RESET => RESET,
      EdgeRESET => DONE_LOAD
);

    U2 : DoubleEdgeSR                   --generisanje LOAD_NEW1
port map(
      SRout => LOAD_NEW1,
      EdgeSET => GATE_IN,
      RESET => RESET,
        ModeRESET => RESET1,
      EdgeRESET =>DONE_LOAD
);
    U3 : DoubleEdgeSR                   --generisanje LOAD_NEW2 NA POCETKU
port map(
      SRout => SETOUT11,
      EdgeSET => DONE_WRITE,
      RESET => RESET,
        ModeRESET => RESET2,
      EdgeRESET =>DONE_LOAD
);
    U4 : DoubleEdgeSR             --generisanje LOAD_NEW2 posle ivice GATE_IN
port map(
      SRout => SETOUT1,
      EdgeSET => GATE_IN,
      RESET => RESET,
        ModeRESET => RESET2,
      EdgeRESET =>DONE_LOAD
);

    U5 : DoubleEdgeSR       --generisanje LOAD_NEW2 posle zavrsetka brojanja
port map(
      SRout => SETOUT2,
      EdgeSET => LAST_COUNT,
      RESET => RESET,
        ModeRESET => RESET2,
      EdgeRESET =>DONE_LOAD
);

    U6 : DoubleEdgeSR             --generisanje LOAD_NEW3 posle ivice GATE_IN
port map(
      SRout => SETOUT3,
      EdgeSET => GATE_IN,
      RESET => RESET,
        ModeRESET => RESET3,
      EdgeRESET =>DONE_LOAD
);

    U7 : DoubleEdgeSR       --generisanje LOAD_NEW3 posle zavrsetka brojanja
port map(
      SRout => SETOUT4,
      EdgeSET => LAST_COUNT,
        ModeRESET => RESET3,
      RESET => RESET,
      EdgeRESET => DONE_LOAD
);

    U8 : DoubleEdgeSR                   --generisanje LOAD_NEW4
port map(
      SRout => LOAD_NEW4,
      EdgeSET => DONE_WRITE,
        ModeRESET =>RESET4,
      RESET => RESET,
      EdgeRESET => DONE_LOAD
);

    U9 : DoubleEdgeSR       --izlaz iz brojaca je '1' ili FULL_COUNT u modu3
port map(
      SRout => SETOUT,
      EdgeSET => notGATE_IN,
        ModeRESET => WRITE_MODEi,
```

```
      RESET => RESET,
      EdgeRESET => LOAD_NEW3
  );

      U10 : DoubleEdgeSR                          --generisanje LOAD_NEW5
  port map(
      SRout => LOAD_NEW5,
      EdgeSET => GATE_IN,
      RESET => RESET,
        ModeRESET => RESET5,
      EdgeRESET =>DONE_LOAD
  );
      U11 : DoubleEdgeSR--generisanje komande za lecovanje izlaznih registara
  port map(
      SRout => COUNTER_LATCH_COMMAND,
      EdgeSET => notCOUNTER_LATCH,
        ModeRESET => notRESET,
      RESET => RESET,
      EdgeRESET => notREADED_COUNTER
  )
      U12 : DoubleEdgeSR   --citanje brojaca (setuje se sa FE RD_LSB,resetuje se
  port map(                                     sa FE RD_MSB)
      SRout => notREADED_COUNTER,
      EdgeSET => notRD_LSB,
        ModeRESET =>notRESET,
      RESET => RESET,
      EdgeRESET => notRD_MSB
  );

      U13 : DoubleEdgeSR                  --generisanje signala INICIAL_MODE
  port map(
      SRout => INICIAL_MODE,
      EdgeSET => notWRITE_MODE,
        ModeRESET =>notRESET,
      RESET => RESET,
      EdgeRESET => notSTART_LOAD
  );
      U14 : FallingEdgeSR                 --generisanje signala LOADED_NEW
  port map(
      SRout => LOADED_NEW,
      EdgeSET => WRITE_VALUEi,
      RESET => RESET,
      EdgeRESET => notSTART_LOAD
  );

      notSTART_LOAD <= not START_LOAD;
      notCOUNTER_LATCH <= (not COUNTER_LATCH);
      READED_COUNTER <= not (notREADED_COUNTER);
      notGATE_IN<=not GATE_IN;
      notRD_LSB<=not RD_LSB;
      notRD_MSB<=not RD_MSB;
      notRESET<= not RESET;
      notWRITE_MODE<= not WRITE_MODEi;

process(SEL,RW1RW0,WRITE_MODEi,WRITE_VALUEi,RW_MODE,LOADED_LSB,START_LOAD,LOADED
_COUNTER)
begin
--upisivanje u CR registre
      case RW1RW0 is
            when "01" =>                          --samo LSB
            WRITE_LSBi<=WRITE_VALUEi;
            WRITE_ZEROES_MSBi<=WRITE_VALUEi;
            DONE_WRITE<=WRITE_VALUEi;

            when "10" =>                          --samo MSB
            WRITE_MSBi<=WRITE_VALUEi;
            WRITE_ZEROES_LSBi<=WRITE_VALUEi;
            DONE_WRITE<=WRITE_VALUEi;

            when "11" =>                          --prvo LSB pa MSB
            if SEL='1' then
                  if WRITE_VALUEi='0' then
                        if LOADED_LSB='0'then
```

```
                              WRITE_LSBi<='0';        --ucitavanje LSB
                              LOADED_COUNTER<='0';
                        else
                              WRITE_MSBi<='0';        --ucitavanje MSB
                              LOADED_COUNTER<='1';
                        end if;
                  else  --vracanje na neaktivno stanje(1)
                        WRITE_LSBi<='1';
                        WRITE_ZEROES_MSBi<='1';
                        WRITE_MSBi<='1';
                        WRITE_ZEROES_LSBi<='1';
                  end if;
                  DONE_WRITE<=LOADED_COUNTER and WRITE_VALUEi;
            end if;

            when others => null;
      end case;

      if SEL='1' then                     --upisivanje moda
            if    WRITE_MODEi='0' then
                  BINBCD<=RW_MODE(0);
                  MODE<=RW_MODE(3 downto 1);
                  RW1RW0<=RW_MODE(5 downto 4);
                  LOADED_LSB<='0';
            elsif WRITE_VALUEi'event and WRITE_VALUEi='1' then
                  LOADED_LSB<=not LOADED_LSB;
            end if;
      end if;
end process;


process(SEL,READi,READED_LSB)       --citanje
begin
      if SEL='1' then
                  if READi='0' then
                        if READED_LSB='0' then--da li je novo citanje
                              RD_LSB<='0';--ocitavanje LSB
                        else
                              RD_MSB<='0';--ocitavanje MSB
                        end if;

                  elsif READi'event and READi='1' then
                        RD_MSB<='1';--vracanje na neaktivno stanje(jedinicu)
                        RD_LSB<='1';--vracanje na neaktivno stanje(jedinicu)
                        READED_LSB<= not READED_LSB;
                  end if;

            end if;
end process;
READ_MSBi<=RD_MSB;
READ_LSBi<=RD_LSB;

--pri citanju, brojac se zaustavlja dok se ne ocita osim ako nije
--pre citanja,stigla komanda za lecovanje registara LR
ENABLE_FROM_READ<=READED_COUNTER or COUNTER_LATCH_COMMAND;

--LR regisri prate trenutno stanje brojaca osim ako nije stigla komanda
--za njihovo lecovanje ili ako nije zabranjeno brojanje
FOLLOW_COUNT<=not COUNTER_LATCH_COMMAND;


--Odredjivanje MOD-a

process(CLK_IN,DONE_WRITE,FULL_COUNT,GATE_IN,MODE,LOADED_NEW,
LOAD_NEW1,LAST_COUNT,ENABLE_FROM_READ,WRITE_VALUEi,
LOAD_NEW1,LOAD_NEW2,LOAD_NEW3,SETOUT,SETOUT2,SETOUT1,SETOUT3,SETOUT4,INICIAL_MOD
E,RW1RW0)
begin
COUNT_CLK<=CLK_IN;

      case MODE is

            when "000" =>     -- mod 0
```

```
      RESET0<='1';RESET1<='0';RESET2<='0';RESET3<='0';RESET4<='0';RESET5<='0';

                  if LOADED_NEW='0' then--dok se spolja upisuje u kolo,
                        COUNTER_OUT<='0';--izlaz  se  postavlja  na  inicijalno
stanje(nulu)
                  end if;

                  if FULL_COUNT='1' then
                        COUNTER_OUT<='1';
                  end if;
                  ENABLE_COUNT<=GATE_IN and ENABLE_FROM_READ and DONE_WRITE and
(INICIAL_MODE nand (RW1RW0(1) or RW1RW0(0)));


      when "001" =>--mod 1

      RESET0<='0';RESET1<='1';RESET2<='0';RESET3<='0';RESET4<='0';RESET5<='0';
            if FULL_COUNT='1' then
                  COUNTER_OUT<='1';--izlaz    se    postavlja    na    inicijalno
stanje(jedinicu)
            end if;

            if  LOAD_NEW1='1' then
                  COUNTER_OUT<='0';--promena stanja
            end if;

            ENABLE_COUNT<=ENABLE_FROM_READ and DONE_WRITE and (INICIAL_MODE nand
(RW1RW0(1) or RW1RW0(0)));

      when "010" =>--mod 2

      RESET0<='0';RESET1<='0';RESET2<='1';RESET3<='0';RESET4<='0';RESET5<='0';
            ENABLE_COUNT<=GATE_IN  and  ENABLE_FROM_READ  and  (INICIAL_MODE  nand
(RW1RW0(1) or RW1RW0(0)));

            if LOADED_NEW='0' then
                  COUNTER_OUT<='1';--izlaz    se    postavlja    na    inicijalno
stanje(jedinicu)
            else
                  COUNTER_OUT<= not LAST_COUNT;
            end if;

      when "011" =>--mod 3

      RESET0<='0';RESET1<='0';RESET2<='0';RESET3<='1';RESET4<='0';RESET5<='0';
            COUNTER_OUT<= (SETOUT or FULL_COUNT) and (not INICIAL_MODE);
            ENABLE_COUNT<=ENABLE_FROM_READ and GATE_IN  and (INICIAL_MODE nand
(RW1RW0(1) or RW1RW0(0)));

      when "100" =>--mod 4

      RESET0<='0';RESET1<='0';RESET2<='0';RESET3<='0';RESET4<='1';RESET5<='0';
            if DONE_WRITE='0' and INICIAL_MODE='0' then
                  COUNTER_OUT<='1';--izlaz se postavlja na inicijalno stanje(1)
            else
                  COUNTER_OUT<=not FULL_COUNT;
            end if;
            ENABLE_COUNT<=GATE_IN  and  ENABLE_FROM_READ  and  (INICIAL_MODE  nand
(RW1RW0(1) or RW1RW0(0)));

      when "101" =>--mod 5

RESET0<='0';RESET1<='0';RESET2<='0';RESET3<='0';RESET4<='0';RESET5<='1';
            if DONE_WRITE='0' and INICIAL_MODE='0' then
                  COUNTER_OUT<='1';--izlaz se postavlja na inicijalno stanje(1)
            else
                  COUNTER_OUT<=not FULL_COUNT;
            end if;
            ENABLE_COUNT<=ENABLE_FROM_READ and (INICIAL_MODE nand (RW1RW0(1) or
RW1RW0(0)));

            when others=>null;
```

```
      end case;
end process;

LOAD_NEW2<=SETOUT1 or SETOUT2 or SETOUT11;
LOAD_NEW3<=SETOUT3 or SETOUT4;
START_LOAD<=LOAD_NEW0 or LOAD_NEW1 or LOAD_NEW2 or LOAD_NEW3 or LOAD_NEW4 or
LOAD_NEW5;
LOAD_COUNTER<=START_LOAD;
COUNT_TWO<=MODE(1) and MODE(0);     --(1) u modu 3,(0) u ostalim modovima

end architecture FSM;
```

MOD 0

U MOD-u 0, nakon učitavanja inicijalne vrednosti u brojač, izlaz brojača se postavlja u stanje logičke nule i u njemu ostaje dok brojač ne odbroji do kraja. Tada prelazi u stanje logičke jedinice i u njemu ostaje do upisivanja novog MOD-a ili nove vrednosti za brojanje.
Brojanje je zabranjeno na početku do upisivanja inicijalne vrednosti, zatim dok je *GATE* u stanju logičke nule i dok se čita stanje brojača bez komande za lečovanje.

MOD 1

U MOD-u 1, nakon upisivanja inicijalne vrednosti za brojanje čeka se rastuća ivica signala *GATE* da bi se učitao brojač i započelo brojanje. Tada izlaz brojača prelazi u stanje logičke nule i u njemu ostaje dok brojač ne stigne do kraja kada prelazi u stanje logičke jedinice. U tom stanju ostaje do upisivanja novog MOD-a ili nove vrednosti za brojanje. Brojanje je zabranjeno na početku do upisivanja inicijalne vrednosti i dok se čita stanje brojača bez komande za lečovanje.

MOD 2

U MOD-u 2, nakon upisivanja inicijalne vrednosti za brojanje izlaz se postavlja u stanje logičke jedinice, a postaje logička nula dok je signal LAST_COUNT na logičkoj jedinici. Brojanje je zabranjeno dok je *GATE* na logičkoj nuli i dok se čita stanje brojača bez komande za lečovanje.

MOD 3

U MOD-u 3, nakon upisivanja inicijalne vrednosti izlaz uzima vrednost signala FULL koji dolazi iz brojača. Učitavanje u brojač se obavlja nakon rastuće ivice signala *GATE* i nakon završetka jednog ciklusa brojanja (RELOAD). Brojanje je zabranjeno dok je *GATE* na logičkoj nuli i dok se čita stanje brojača bez komande za lečovanje.

MOD 4

U MOD-u 4, nakon upisivanja inicijalne vrednosti za brojanje izlaz se postavlja u stanje logičke jedinice, a postaje logička nula kada brojač stigne do kraja i traje jednu periodu CLK-a. Nakon toga postaje logička jedinica i tako ostaje do upisivanja novog MOD-a ili nove vrednosti za brojanje.
Brojanje je zabranjeno dok je *GATE* na logičkoj nuli i dok se čita stanje brojača bez komande za lečovanje.

MOD5

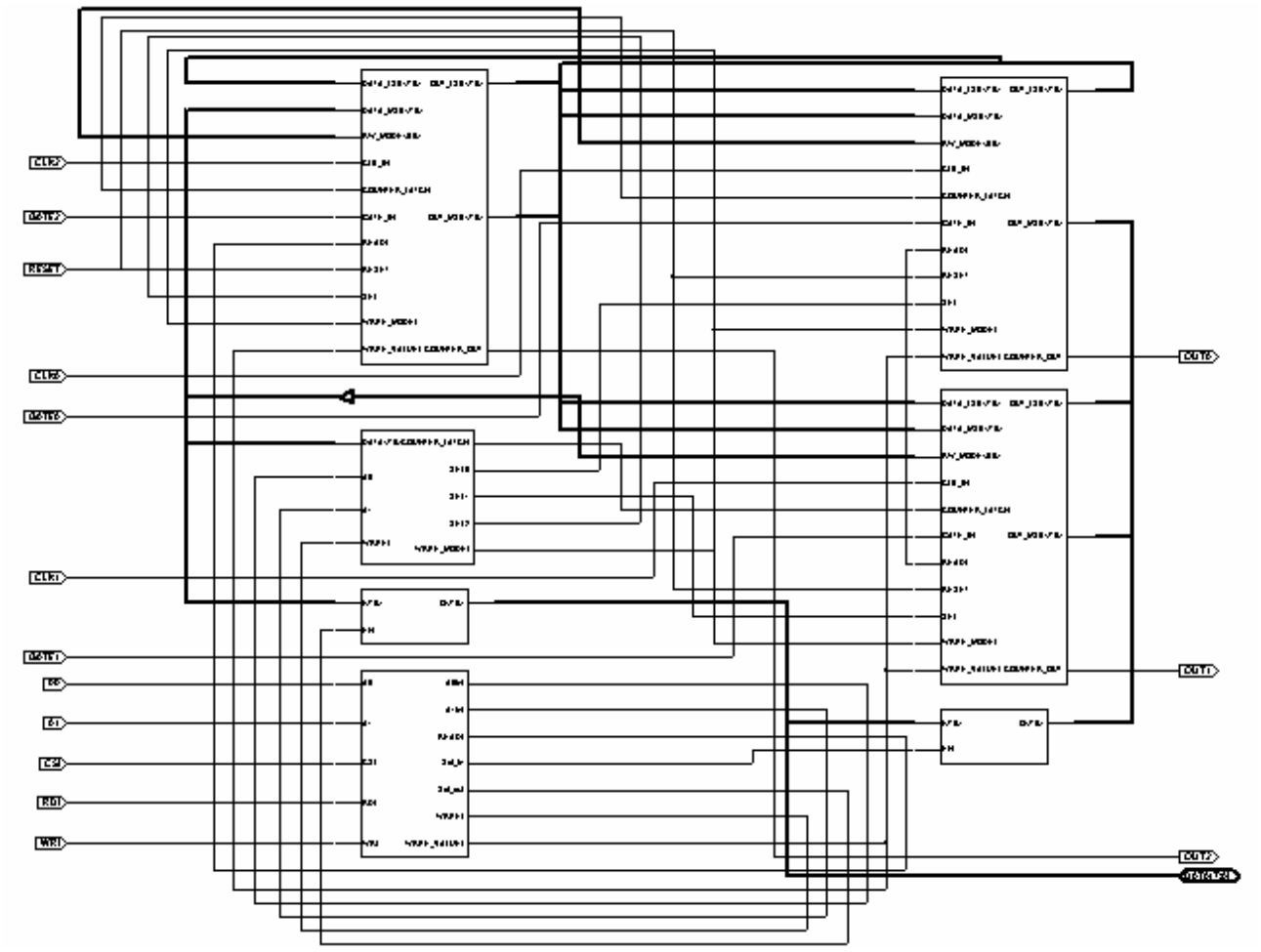
U MOD-u 5, nakon upisivanja inicijalne vrednosti za brojanje, čeka se rastuća ivica signala *GATE* da bi se učitala vrednost u brojač i da bi započelo brojanje. Izlaz se postavlja u stanje logičke jedinice i takav ostaje do završetka brojanja, kada postaje nula jednu periodu CLK-a, a zatim se vraća u stanje logičke jedinice i takav ostaje do upisivanja novog MOD-a ili nove vrednosti za brojanje. Brojanje je zabranjeno dok je *GATE* na logičkoj nuli i dok se čita stanje brojača bez komande za lečovanje.

4. SINTEZA I IMPLEMENTACIJA KOLA

Kao što je na početku opisa rečeno kolo je sintetizovano u programskom paketu firme *XILINX* ISE6.3i. Ovo je programski paket firme *XILINX* za razvoj aplikacija baziranih na njihovim CPLD i FPGA kolima.

Za implementaciju programabilnog interval tajmera 8253 iskorišćeno je FPGA kolo iz *XILINX*-ove familije SPARTAN2 koje nosi oznaku XC2S30-VQ100. Radi se o kolu koje ima 30K gejtova i ima ugrađene periferije poput PCI intefejsa, AGP 2x interfejsa itd. Dolazi u 100 pinskom VQ (very thin quad flat pack) kućištu. Upotrebljeno je iako nije bilo potrebe za tolikim brojem pinova, jer je to najmanje pakovanje u okviru pomenute serije . Kolo koristi napajanje od 2.5V.

Na slici 17. prikazana je šema sinteze kola na najvišem nivou, koju je izgenerisao program Xilinx ECS. Na šemi se mogu uočiti osnovni blokovi od kojih se kolo sastoji, gde se naravno u programu može analizirati svaki blok dubinski do nivoa samih gejtova.



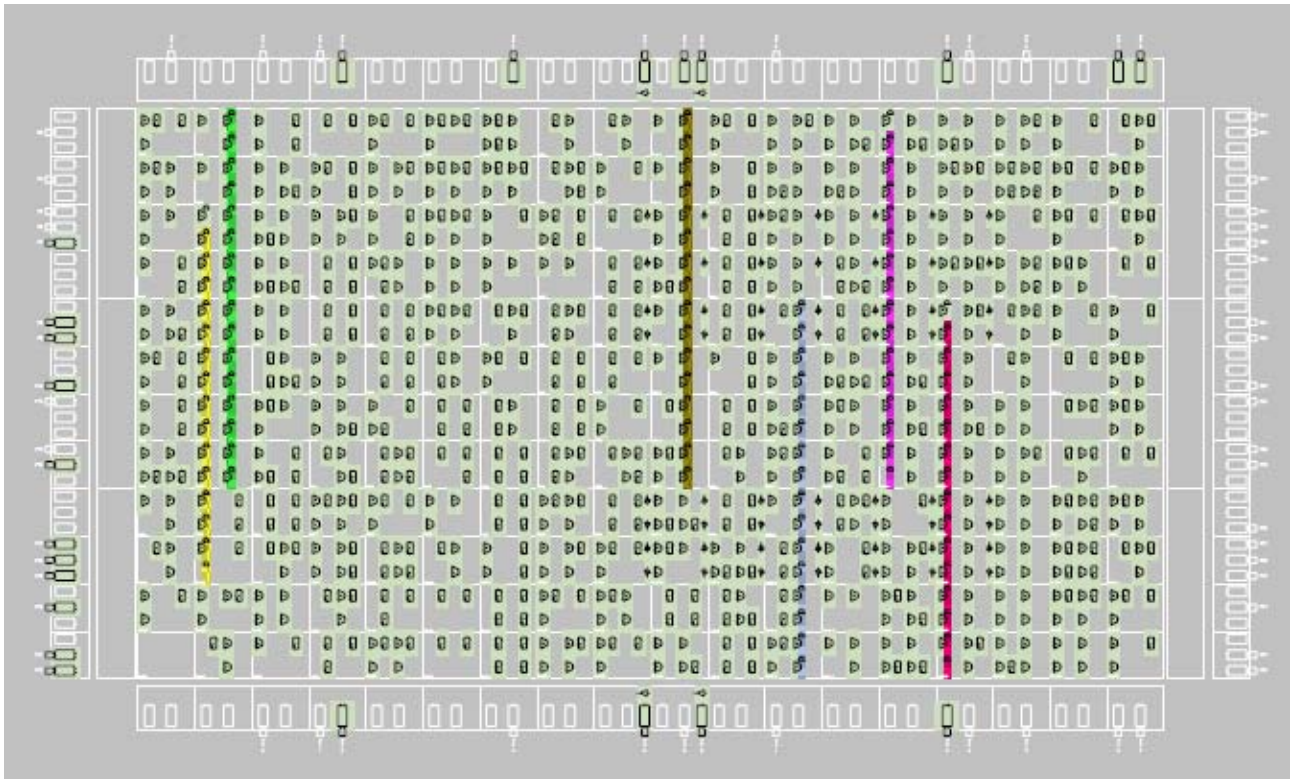
Slika 17. Šema sinteze

Nakon toga prešlo se na samu implementaciju kola na konkretnom FPGA kolu. Sledeći listing nam pokazuje dobijene rezultate:

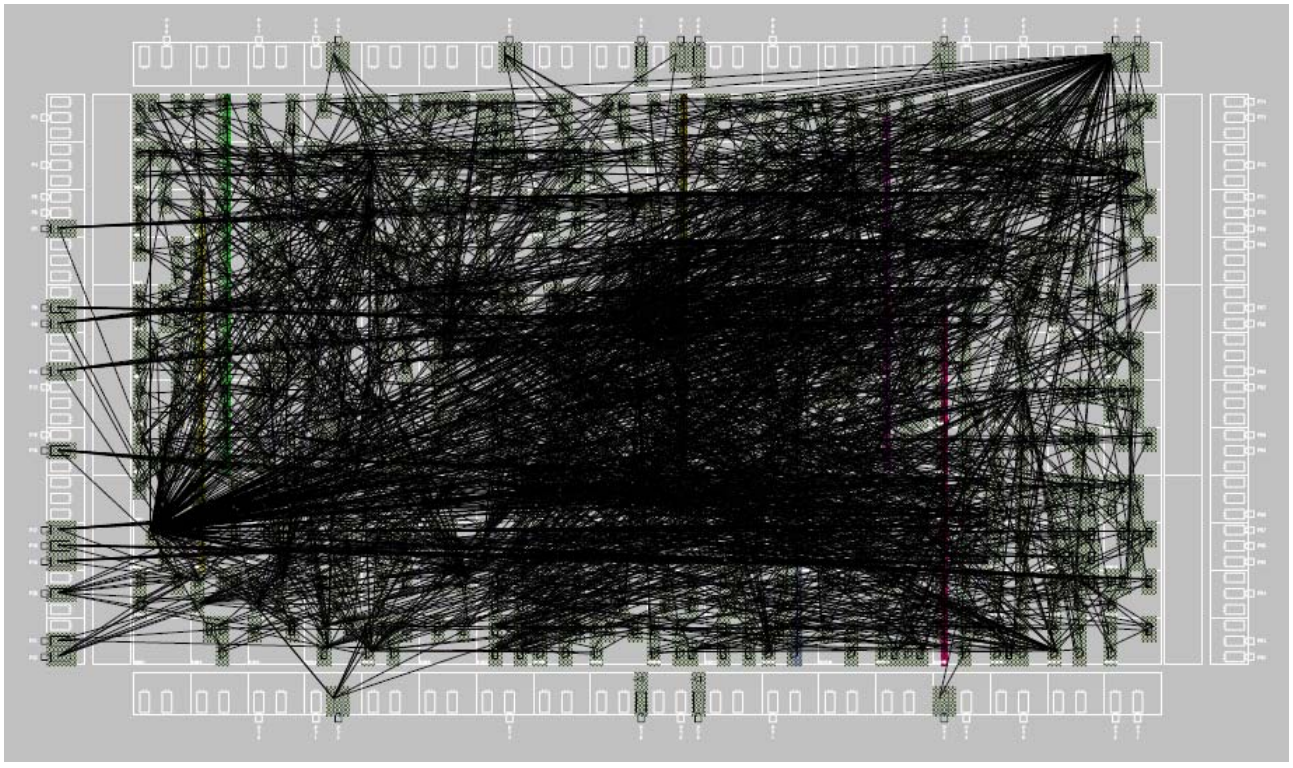
```
Design Information
------------------
Command Line   : E:/Xilinx/bin/nt/map.exe -intstyle ise -p xc2s30-vq100-6 -cm
area -pr b -k 4 -c 100 -tx off -o pit8253_map.ncd pit8253.ngd pit8253.pcf
Target Device  : x2s30
Target Package : vq100
Target Speed   : -6
Mapper Version : spartan2 -- $Revision: 1.16.8.2 $
Mapped Date    : Mon May 23 01:17:51 2005

Design Summary
--------------
Number of errors:      0
Number of warnings:   80
Logic Utilization:
  Total Number Slice Registers:     353 out of    864   40%
    Number used as Flip Flops:              138
    Number used as Latches:                 215
  Number of 4 input LUTs:          554 out of    864   64%
Logic Distribution:
    Number of occupied Slices:                    430 out of    432   99%
    Number of Slices containing only related logic:    386 out of    430   89%
    Number of Slices containing unrelated logic:        44 out of    430   10%
       *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:         557 out of    864   64%
      Number used as logic:                554
      Number used as a route-thru:           3
   Number of bonded IOBs:           19 out of     60   31%
      IOB Latches:                        3
   Number of Tbufs:                56 out of    480   11%
   Number of GCLKs:                 4 out of      4  100%
   Number of GCLKIOBs:              4 out of      4  100%

Total equivalent gate count for design:  6,385
Additional JTAG gate count for IOBs:  1,104
```

Na osnovu dobijenih podataka za nas je bitno da se na kraju vidi da je implemetnirano kolo zauzelo 6385 ekvivalentnih gejtova, kao i da je pridodato još 1104 gejta koja su neophodna za realizaciju JTAG logike kojom se vrši testiranje i programiranje FPGA kola. Kao još jedan važan podatak treba istaći da je maksimalna radna frekvencija kola 114.051MHz.

Na slikama 18a. i 18b. vidimo unutrašnju strukturu dobijenu nakon implementacije kola gde se vide zauzeti blokovi unutar kola. Slike su uzete iz *Xilinx Floorplanner* editora koji je zadužen u okviru paketa ISE za sam fizički razmeštaj ćelija. Slika 18a. prikazuje sam razmeštaj ćelija gde možemo videti i koji pinovi kola su iskorišćeni. Iako se zapaža da izvodi kola nisu grupisani, to je da bi se postiglo približno isto kašnjenje veza do svih pinova. Izgled povezanosti samih veza prikazan je na slici 18b.



Slika 18a.Šema zauzetosti ćelija FPGA kola

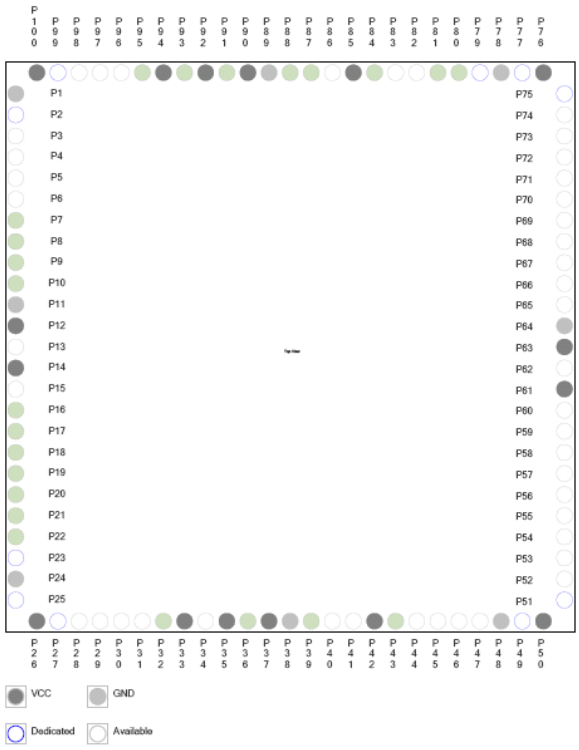


Slika 18b.Šema veza ćelija FPGA kola

Na kraju izvršena je i analiza potrošnje kola u delu paketa pod imenom *XPower,* je pokazao totalnu potrošnju od 6.6mW. Pri tome pri temperaturi ambijenta od 25 °C, temperatura kućišta je 25,2 °C.

Sledi listing koji prikazuje na kojem pinu FPGA kola su povezani ulazi i izlazi programabilnog interval tajmera 8253. Na slici 19. prikazan je i sam fizički raspored pinova na FPGA kolu.

```
#PINLOCK_BEGIN

NET "RDi"          LOC =  "P22";          NET "GATE2"        LOC =  "P95";
NET "WRi"          LOC =  "P21";          NET "GATE0"        LOC =  "P93";
NET "CSi"          LOC =  "P20";          NET "GATE1"        LOC =  "P80";
NET "DATA<6>"      LOC =  "P19";          NET "RESET"        LOC =  "P81";
NET "DATA<7>"      LOC =  "P18";          NET "OUT1"         LOC =  "P84";
NET "DATA<5>"      LOC =  "P17";          NET "OUT0"         LOC =  "P43";
NET "DATA<4>"      LOC =  "P16";
NET "DATA<0>"      LOC =  "P10";
NET "DATA<2>"      LOC =  "P9";
NET "DATA<1>"      LOC =  "P8";
NET "DATA<3>"      LOC =  "P7";
NET "A1"           LOC =  "P32";
NET "A0"           LOC =  "P36";
NET "CLK2"         LOC =  "P91";
NET "CLK0"         LOC =  "P39";
NET "CLK1"         LOC =  "P88";
NET "OUT2"         LOC =  "P87";

#PINLOCK_END
```



Slika 19. Raspored pinova FPGA kola

Ovim je završen ceo proces sinteze i implementacije kola. Program generiše i programski fajl kojim je potrebno isprogramirati FPGA kolo da bi se dobila željena funkcija.

5. TESTIRANJE RADA KOLA

U cilju verifikacije dizajna, napisan je *TestBench* program i izvršena je simulacija rada kola. Sva tri brojača su identična pa je dovoljno posmatrati samo jedan. U našem slučaju, izabran je brojač1 i simuliran je njegov rad.

5.1 TESTIRANJE BROJAČA U MOD-u 0:

Proceduru testiranja sprovešćemo u sledećim koracima**:**

- Na početku je potrebno resetovati kolo, što se vrši postavljanjem signala RESET na logičku jedinicu u periodu 5 μs do 10 μs, nakon čega se vraća na neaktivno stanje (logičku nulu) do kraja intervala testiranja.
- Nakon ovog koraka je potrebno inicijalizirati brojač tj. upisati kontrolnu reč i vrednost brojanja. Prvo se adresira *Control Word* registar postavljanjem A1='1' i A0='1', u periodu od 15 μs do 35 μs, nakon čega se signali A1 i A0 vraćaju u neaktivno stanje visoke impedanse
- U istom periodu se postavlja *Control Word* koji odredjuje MOD brojača (DATA="01 11 000 0", za detalje vidi Tabele 2-6 u Sekciji **Opis Rada**)
- U periodu od 20 μs do 30 μs upisuje se *Control Word* postavljanjem signala WRi u aktivno stanje (logička nula). Istovremeno se postavlja i signal CSi u aktivno stanje (logička nula).
- Adresira se brojač1 postavljanjem adresnih linija A1 = '0' i A0 = '1', a na magistralu se postavlja LSB podatak (vrednost 5) u 45 μs
- Vrši se upis u brojač postavljanjem signalaWRi i CSi u aktivna stanja (logičke nule) u periodu od 50 μs do 60 μs
- Postavlja se MSB podatak na magistaralu (vrednost 0) u 65 μs
- Upisuje se MSB podatka postavljanjem signalaWRi i CSi u stanje logičke nule u periodu od 70 μs do 80 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA), u 85 μs se postavljaju u stanje visoke impedanse

Ovim je završena inicijalizacija brojača, i na prvu sledeću opadajuću ivicu CLK-a (96 μs) obaviće se učitavanje inicijalne vrednosti u brojač, i brojač počinje da broji. Izlaz brojača je u stanju logičke nule. Karakteristične aktivnosti u toku brojanja su sledeće:

- U 160 μs postaviti signal *GATE1* na logičku nulu što dovodi do zaustavljanja brojanja
- U 200 μs *GATE1* vratiti u stanje logičke jedinice što uzrokuje nastavak brojanja
- Kada brojač odbroji do kraja (296 μs), izlaz brojača prelazi u stanje logičke jedinice i u njemu ostaje do upisivanja novog MOD-a, ili nove vrednosti za brojanje

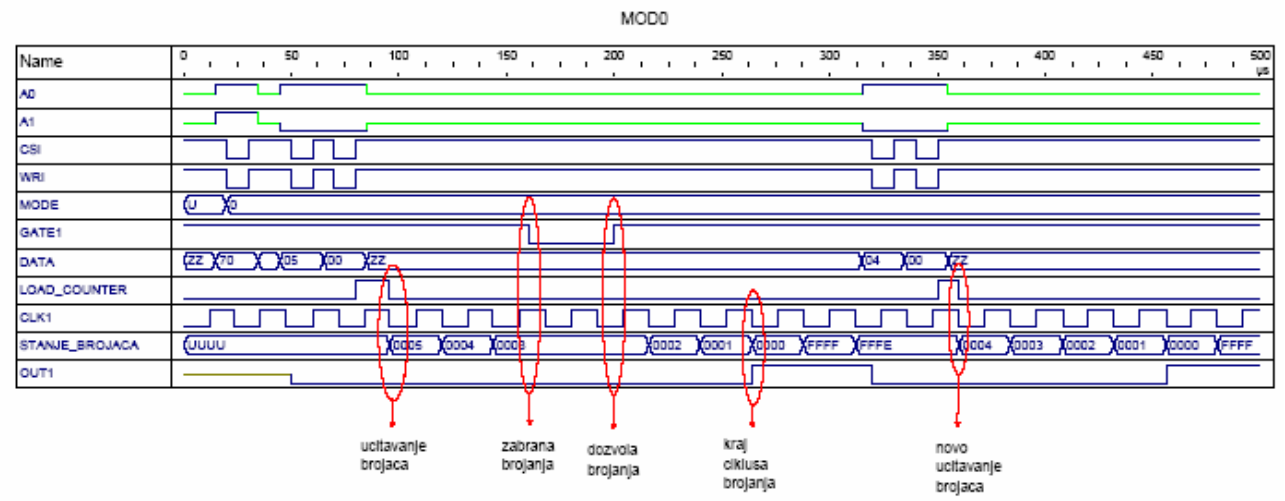
Upisati novu vrednost za brojanje na sledeći način:

- U 315 μs adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (broj 4).
- U periodu od 320 μs do 330 μs postaviti signale WRi i CSi u stanje logičke nule što izaziva upis LSB podatka
- U trenutku 335 μs postaviti na magistaralu MSB podatak (broj 0).
- U periodu od 340 μs do 350 μs postaviti signale WRi i CSi u aktivna stanja što dovodi do upisa MSB podatka (teži deo modula brojanja)
- U 355 μs adresne linije (A1 i A0) kao i linije na magistrali (DATA) postaviti u neaktivno stanje, tj. stanje visoke impedanse

Nakon upisa drugog bajta, na prvu sledeću opadajuću ivicu CLK-a (360 μs) obaviće se učitavanje nove vrednosti u brojač i brojač počinje da broji. Izlaz brojača je u stanju logičke nule.

Kada brojač odbroji do kraja (456 μs) izlaz brojača prelazi u stanje logičke jedinice i u tom stanju ostaje do upisivanja novog MOD-a, ili nove vrednosti za brojanje.

Dobijeni vremenski dijagram prikazan je na sledećoj slici:



Slika 17. Vremenski dijagram rada brojača u MOD-u 0

5.2 TESTIRANJE BROJAČA U MOD-u 1:

- Na početku je potrebno resetovati kolo tako što će se postaviti signal RESET u aktivno stanje (logička jedinica) u periodu od 5 μs do 10 μs, a zatim treba signal RESET vratiti u neaktivno stanje (logičku nulu) do kraja testiranja
- Adresirati *CONTROL WORD* registar postavljanjem adresnih linija A1 = '1' i A0 = '1' u 15 μs
- Postaviti *Control Word* koji određuje MOD brojača (DATA="01 01 001 0") u 15 μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u aktivna stanja (logičke nule) u periodu od 20 μs do 30 μs.
- Adresne linije (A1 i A0) i linije na magistrali (DATA), u 35 μs se postavljaju u stanje visoke impedanse
- Adresirati brojač1 postavljanjem adresnih linija A1 = '0' i A0 = '1' i postaviti na magistralu LSB podatak (vrednost 5) u 65 μs
- Upisati LSB podatak postavljanjem signala WRi i CSi u periodu od 70 μs do 80 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA), postaviti u stanje visoke impedanse u 85 μs

Time je završena inicijalizacija i za početak brojanja je potrebna rastuća ivica na ulazu *GATE1*.

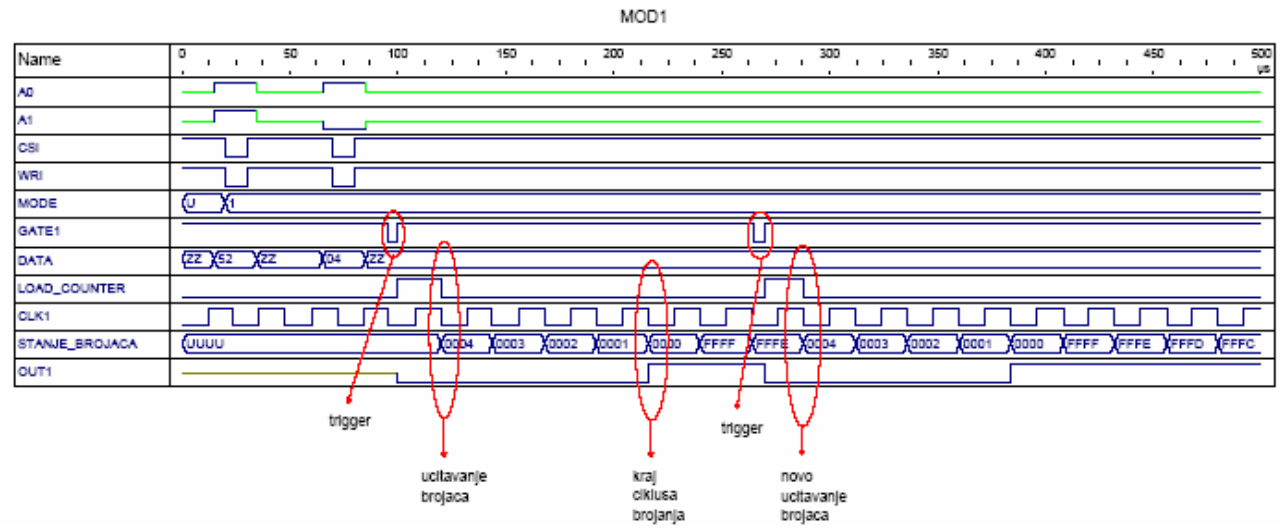
- Postaviti signal GATE1 u stanje logičke nule u 95 μs, a zatim u stanje logičke jedinice u 100 μs. Time se dobija tražena rastuća ivica koja izaziva učitavanje stanja u brojač na prvu sledeću opadajuću ivicu CLK-a i početak brojanja. Izlaz brojača se postavlja u stanje logičke nule.

Kada brojač odbroji do kraja (216 μs) izlaz brojača prelazi u stanje logičke jedinice i u njemu ostaje do upisivanja novog MOD-a ili nove vrednosti za brojanje ili do pojave nove rastuće ivice ulaznog signala *GATE1*.

- Generisati novu rastuću ivicu signala GATE1 tako što će se postaviti u stanje logičke nule u 265 μs, a zatim u stanje logičke jedinice u 270 μs

Na prvu sledeću opadajuću ivicu CLK-a (288 μs) obaviće se učitavanje brojača i počinje brojanje. Izlaz brojača se postavlja u stanje logičke nule i u njemu ostaje dok brojač ne odbroji do kraja.
U 384 μs, brojač je odbrojao do kraja i izlaz brojača prelazi u stanje logičke jedinice.

Dobijeni vremenski dijagram prikazan je na sledećoj slici:



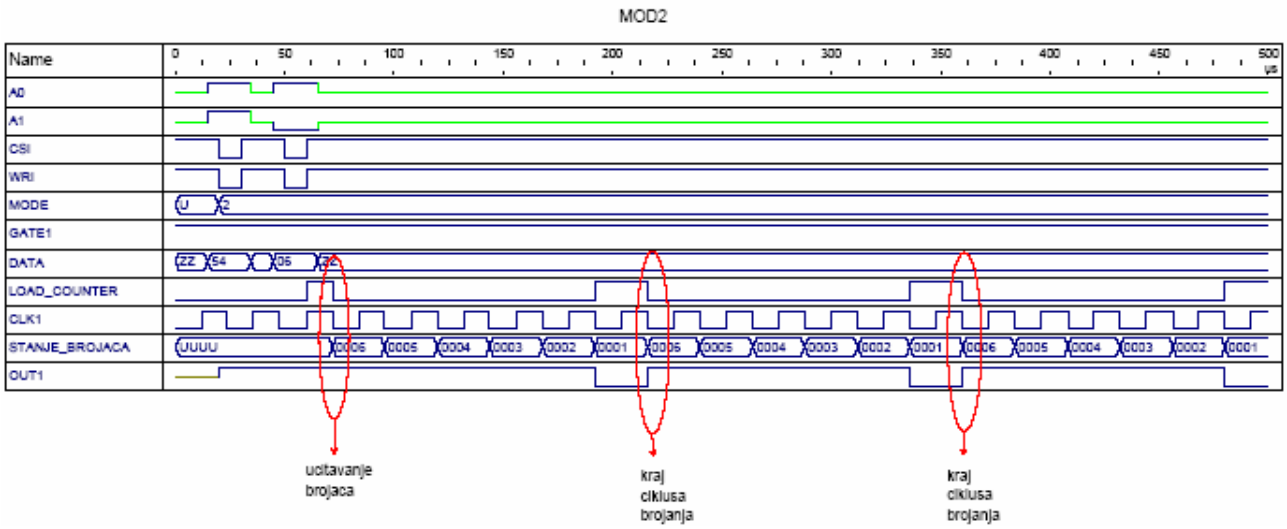
Slika 18. Vremenski dijagram rada brojača u MOD-u 1

5.3 TESTIRANJE BROJAČA U MOD-u 2:

- Resetovati kolo postavljanjem signala RESET u stanje logičke jedinice u periodu od 5 μs do 10 μs, a nakon toga ga treba vratiti u neaktivno stanje (logičku nulu) do kraja testiranj.
- Adresirati *CONTROL WORD* registar postavljanjem signala A1 = '1' i A0 = '1' u 15 μs
- Postaviti *Control Word* koji sadrži MOD brojača (DATA="01 01 010 0") u 15 μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u stanje logičke nule u periodu od 20 μs do 30 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 35 μs
- Adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 6) u 45μs
- Upisuje se LSB podatak aktiviranjem signala WRi i CSi u periodu 50 μs do 60 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 65 μs

Time je završena inicijalizacija i nakon pojave opadajuće ivice CLK-a obaviće se učitavanje brojača i početak brojanja. Izlaz brojača se postavlja u stanje logičke jedinice.
Kada brojač odbroji do jedinice (stanje "0000 0000 0000 0001" – 216 μs) izlaz brojača perelazi u stanje logičke jedinice i u njemu ostaje jednu periodu CLK-a. Nakon toga se automatski vrši ponovno učitavanje početne vrednosti i počinje novi ciklus brojanja, a izlaz brojača se vraća u stanje logičke jedinice.
Opisani ciklus se ponavlja beskonačno do upisa novog MOD-a.

Dobijeni vremenski dijagram prikazan je na sledećoj slici:



Slika 19. Vremenski dijagram rada brojača u MOD-u 2

5.4 TESTIRANJE BROJAČA U MOD-u 3:

- Resetovati kolo postavljanjem signala RESET u stanje logičke jedinice u periodu od 5 μs do 10 μs, a nakon toga ga treba vratiti u neaktivno stanje (logičku nulu) do kraja testiranja
- Adresirati *CONTROL WORD* registar postavljanjem signala A1 = '1' i A0 = '1' u 15 μs
- Postaviti *Control Word* koji sadrži MOD brojača (DATA="01 01 010 0") u 15 μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u stanje logičke nule u periodu od 20 μs do 30 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 35 μs
- Adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 5) u 45μs
- Upisuje se LSB podatak aktiviranjem signala WRi i CSi u periodu 50 μs do 60 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 65 μs

Time je završena inicijalizacija i za početak brojanja je potrebna rastuća ivica na ulazu *GATE1*.

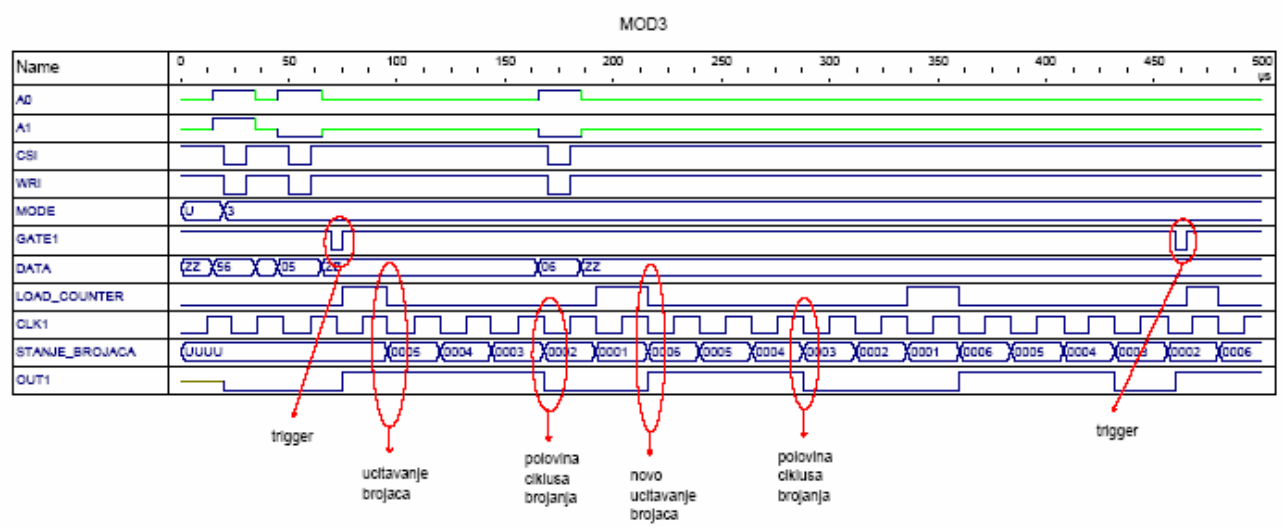
- Postaviti signal GATE1 u stanje logičke nule u 70 μs, a zatim u stanje logičke jedinice u 75 μs. Time se dobija tražena rastuća ivica koja izaziva učitavanje brojača na prvu sledeću opadajuću ivicu CLK-a i početak brojanja. Izlaz brojača se postavlja u stanje logičke jedinice.

Kada brojač odbroji do polovine vrednosti upisane u brojač (168 μs), izlaz brojača prelazi u stanje logičke nule i u njemu ostaje do kraja tog ciklusa brojanja (216 μs). Nakon toga se automatski učitava inicijalna vrednost za brojanje smeštena u ulazne registre brojača, i takav ciklus se ponavlja beskonačno do upisivanja novog MOD-a.

Ako se u toku brojanja upiše nova inicijalna vrednost za brojanje, ona će biti učitana nakon rastuće ivice na ulazu *GATE1* ili na kraju tekućeg ciklusa brojanja.

- Adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 6) u 165 μs
- Upisati LSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 170 μs do 180μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 185 μs

Upisana vrednost biće učitana u brojač nakon završetka tekućeg ciklusa brojanja (216 μs).
Ako se pojavi rastuća ivica na ulazu *GATE1* obaviće se novo učitavanje brojača i početak brojanja, a izlaz brojača se odmah postavlja u stanje logičke jedinice (465 μs).
Dobijeni vremenski dijagram prikazan je na sledećoj slici:



Slika 20. Vremenski dijagram rada brojača u MOD-u 3

5.5 TESTIRANJE BROJAČA U MOD-u 4:

- Resetovati kolo postavljanjem signala RESET u stanje logičke jedinice u periodu od 5 μs do 10 μs, a nakon toga ga treba vratiti u neaktivno stanje (logičku nulu) do kraja testiranja
- Adresirati *CONTROL WORD* registar (A1 = '1', A0 = '1') u 15 μs
- Postaviti *Control Word* koji sadrži MOD brojača (DATA="01 11 100 0") u 15μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u aktivna stanja u periodu od 20μs do 30 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 35 μs
- Adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 5) u 45 μs
- Upisati LSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 50 μs do 60 μs
- Postaviti na magistaralu MSB podatak (broj 0) u 65 μs
- Upisati MSB podatak u perioodu od 70 μs do 80 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 85 μs

Završena je inicijalizacija brojača i na prvu sledeću opadajuću ivicu CLK-a (96 μs) obaviće se učitavanje inicijalne vrednosti u brojač i brojač počinje da broji. Izlaz brojača je u stanju logičke jedinice.
Kada brojač odbroji do kraja (216 μs) izlaz brojača prelazi u stanje logičke nule i u njemu ostaje jednu periodu CLK-a, nakon čega se vraća u stanje logičke jedinice i u njemu ostaje do upisivanja nove inicijalne vrednosti za brojanje ili upisivanja novog MOD-a.

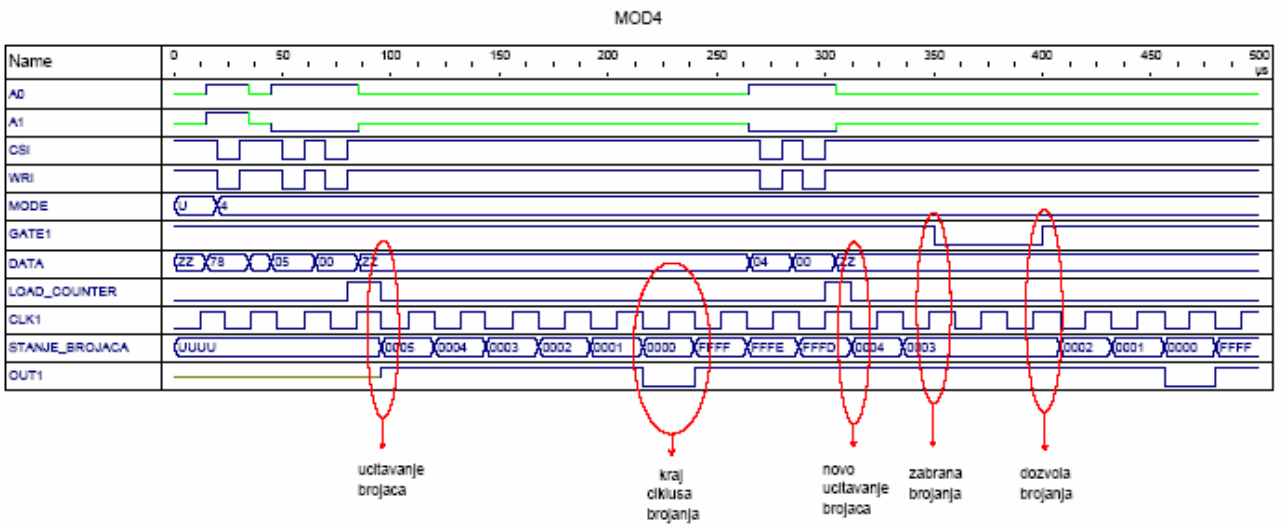
Upisati novu vrednost za brojanje na sledeći način:

- Adresirati brojač1 spostavljanjem adesnih linija (A1 = '0', A0 = '1') i postaviti na magistralu novi LSB podatak (vrednost 4) u 265 μs
- Upisati novi LSB podatak postavljanjem signala WRi i CSi u periodu od 270 μs do 280 μs
- Postaviti na magistaralu novi MSB podatak (vrednost 0) u 285 μs
- Upisati novi MSB podatak u periodu od 290 μs do 300 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 305 μs

Na prvu sledeću opadajuću ivicu CLK-a (312 μs) obaviće se učitavanje nove inicijalne vrednosti za brojanje, a izlaz brojača se nalazi u stanju logičke jedinice do završetka brojanja.

- Postaviti signal *GATE1* u stanje logičke nule u 350 μs čime se zabranjuje brojanje
- Postaviti signal *GATE1* u stanje logičke jedinice u 400 μs čime se dozvoljava dalje brojanje

Dobijeni vremenski dijagram prikazan je na sledećoj slici:



Slika 21. Vremenski dijagram rada brojača u MOD-u 4

5.6 TESTIRANJE BROJAČA U MOD-u 5:

- Resetovati kolo postavljanjem signala RESET u stanje logičke jedinice u periodu od 5 μs do 10 μs, a nakon toga ga treba vratiti u neaktivno stanje (logičku nulu) do kraja testiranja
- Adresirati *CONTROL WORD* registar (A1 = '1', A0 = '1') u 15 μs
- Postaviti *Control Word* koji sadrži MOD brojača (DATA="01 11 101 0") u 15μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u aktivna stanja (logičke nule) u periodu od 20 μs do 30 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 35 μs
- Adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 5) u 45 μs

- Upisati LSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 50μs do 60μs
- Postaviti na magistaralu MSB podatak (vrednost 0) u 65 μs
- Upisati MSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 70 μs do 80 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 85 μs

Završena je inicijalizacija brojača i nakon pojave rastuće ivice na ulazu *GATE1* obaviće se učitavanje brojača i početak brojanja.

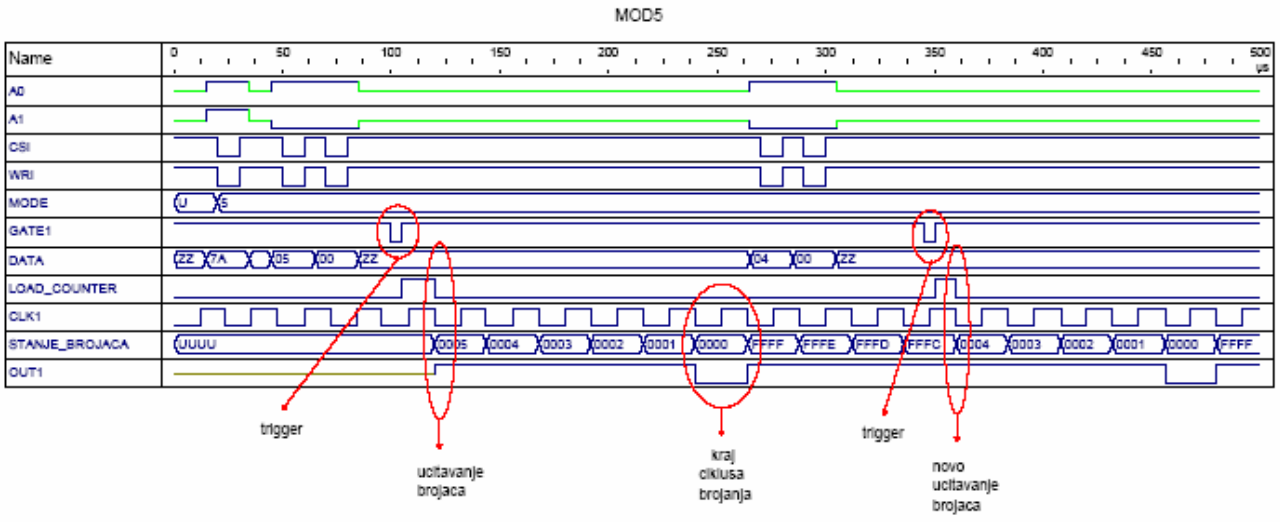
- Postaviti signal GATE1 u stanje logičke nule u 100 μs
- Postaviti signal GATE1 stanje logičke jedinice u 105 μs

Time je na ulazu GATE1 dobijena rastića ivica signala što izaziva učitavanje vrednosti u brojač i početak brojanja. Izlaz brojača se postavlja u stanje logičke jedinice.
Kada brojač odbroji do kraja (240 μs) izlaz brojača perelazi u stanje logičke nule i u njemu ostaje jednu periodu CLK-a, nakon čega se vraća u stanje logičke jedinice i u njemu ostaje do upisivanja nove inicijalne vrednosti za brojanje, novog MOD-a ili nove rastuće ivice ulaznog signala *GATE1*.

Upisati novu vrednost za brojanje na sledeći način:

- Adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 4) u 265 μs
- Upisati LSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 270 μs do 280 μs
- Postaviti na magistaralu MSB podatak (vrednost 0) u 285 μs
- Upisati MSB podatak u periodu od 290 μs do 300 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 305 μs

Nakon rastuće ivice ulaznog signala *GATE1* (350 μs), obaviće se očitavanje nove inicijalne vrednosti za brojanje, a izlaz brojača se postavlja u stanje logičke jedinice.
Dobijeni vremenski dijagram prikazan je na sledećoj slici:



Slika 22 Vremenski dijagram rada brojača u MOD-u 5

5.7 TESTIRANJE OČITAVANJA BROJAČA :

Za proces očitavanja brojača potrebno je prethodno upisati MOD rada i inicijalnu vrednost za brojanje i startovati brojanje.

- Resetovati kolo postavljanjem signala RESET u stanje logičke jedinice u periodu od 5 μs do 10 μs, a nakon toga ga treba vratiti u neaktivno stanje (logičku nulu) do kraja testiranja
- Adresirati *CONTROL WORD* registar (A1 = '1', A0 = '1') u 15 μs
- Postaviti *Control Word* koji sadrži MOD brojača (DATA="01 11 000 0") u 15μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u aktivna stanja (logičke nule) u periodu od 20 μs do 30 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 35 μs
- adresirati brojač1 (A1 = '0', A0 = '1') i postaviti na magistralu LSB podatak (vrednost 7) u 45 μs
- Upisati LSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 50μs do 60μs
- Postaviti na magistaralu MSB podatak (vrednost 0) u 65 μs
- Upisati MSB podatak postavljanjem signala WRi i CSi u aktivna stanja u periodu od 70 μs do 80 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 85 μs

Na sledeću opadajuću ivicu CLK-a učitaće se brojač i počeće da broji.
Očitavanje brojača se obavlja na sledeći način:

- Adresirati brojač1 (A1 = '0', A0 = '1') u 155 μs
- Postaviti spoljne signale RDi i CSi u aktivna stanja (logička nula) u periodu od 160 μs do 170 μs

Prvo aktivno stanje signala RDi izaziva zaustavljanje brojanja i očitavanje LSB podatka.

- Postaviti spoljne signale RDi i CSi u aktivna stanja po drugi put u periodu od 180 μs do 190 μs što izaziva očitavanje MSB podatka

Nakon očitavanja oba bajta, kada se spoljni signal RDi vrati na neaktivno stanje (jedinicu), nastavlja se brojanje brojača.

Drugi način za očitavanje je očitavanje sa prethodnim lečovanjem stanja izlaznih registara pri čemu se u toku očitavanja ne zaustavlja brojanje brojača.
Takva procedura očitavanja se odvija na sledeći način:
- Upisati odgovarajući *Control Word* u *Control Word* registar i tada se stanje izlaznih registara "zamrzava" i ostaje takvo do završetka procesa očitavanja. *Control Word* treba da ima format "01 00 xxxx", pri čemu prva dva bita predstavljanu adresu brojača koji će se očitati a druga dva bita predstavljaju komandu za lečovaje stanja izlaznih registara.

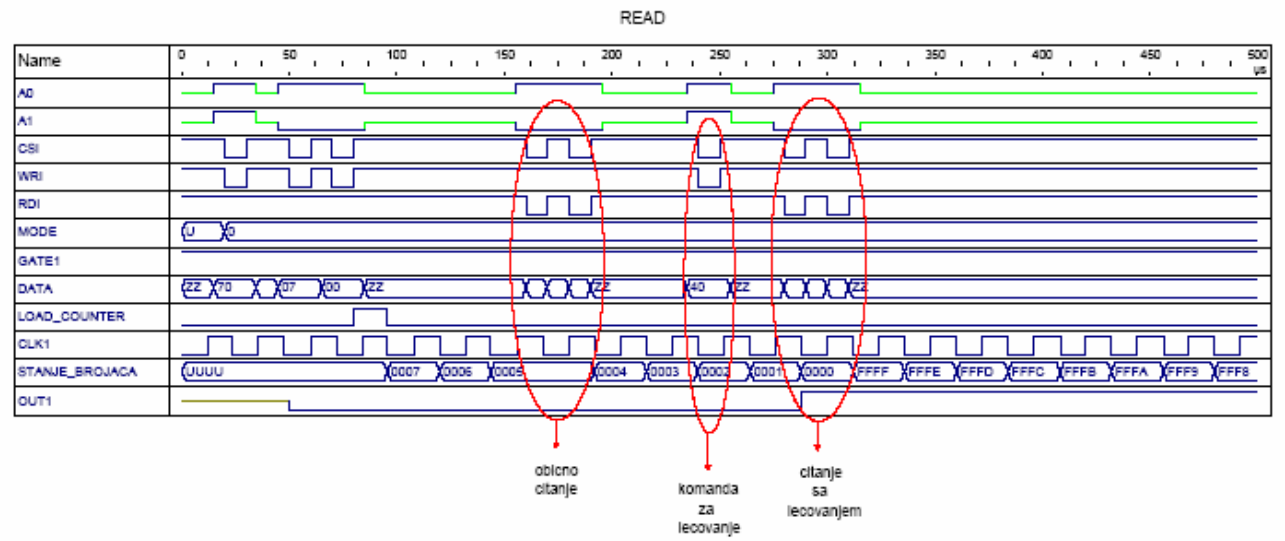
- Adresirati *CONTROL WORD* registar (A1 = '0', A0 = '1') u 235 μs
- Postaviti *Control Word* koji sadrži komandu za lečovanje (DATA="01 00 000 0") u 235μs
- Upisati *Control Word* postavljanjem signala WRi i CSi u aktivna stanja (logičke nule) u periodu od 240 μs do 250 μs
- Adresne linije (A1 i A0) i linije na magistrali (DATA) postaviti u neaktivno stanje visoke impedanse u 255 μs

Očitavanje:

- Adresirati brojač1 (A1 = '0', A0 = '1') u 275 μs
- Postaviti spoljne signale RDi i CSi u aktivna stanja (logička nula) u periodu od 280 μs do 290 μs. Kao rezultat ove akcije očitaće se LSB podatak a da se pri tom ne zaustavi brojanje

- Postaviti spoljne signale RDi i CSi u aktivna stanja po drugi put u periodu od 300 μs do 310 μs što izaziva očitavanje MSB podatka

Dobijeni vremenski dijagram prikazan je na sledećoj slici:



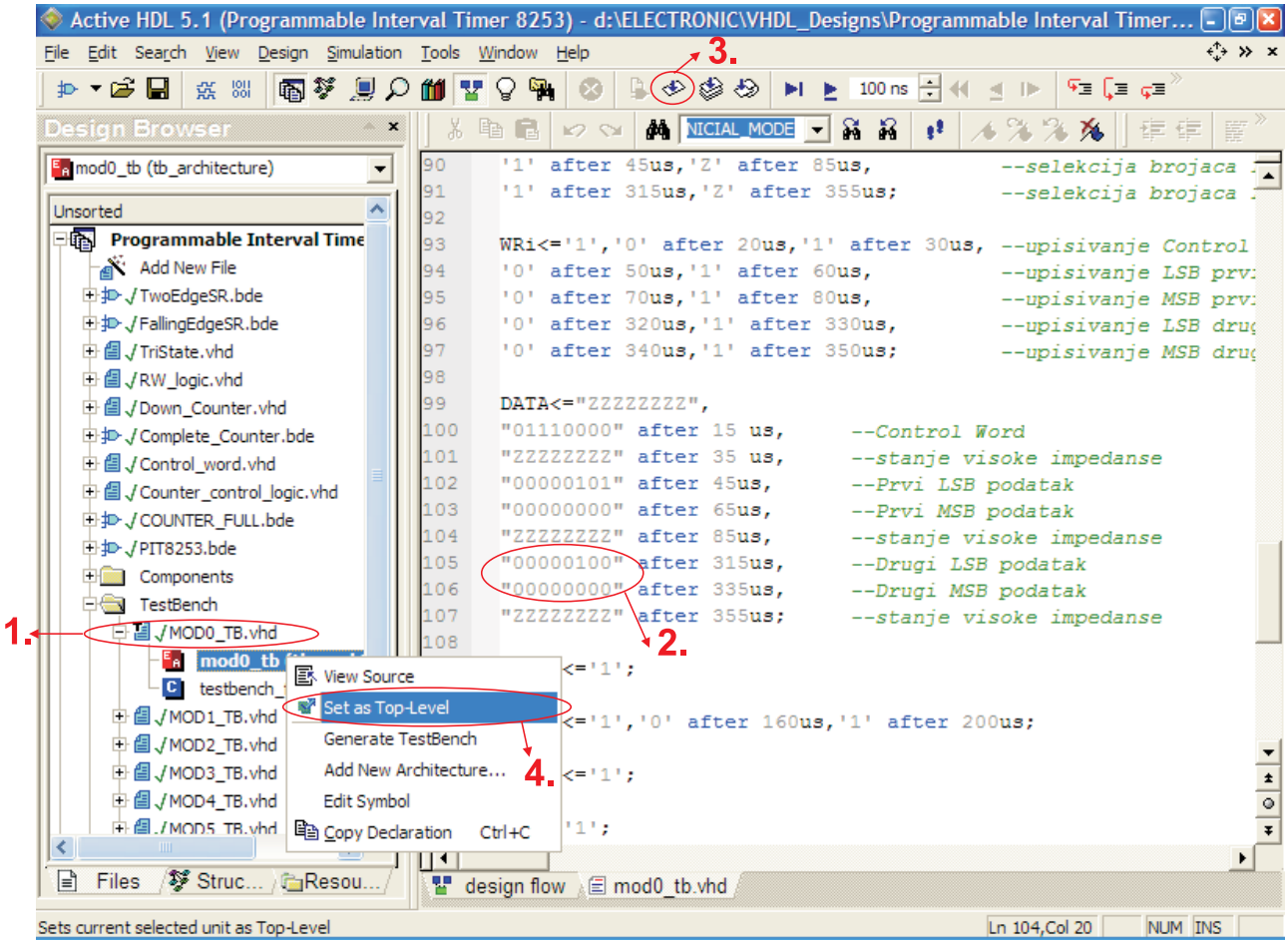
Slika 23. Vremenski dijagram očitavanja brojača

6. ZADATAK:

Proveriti ispravnost rada Tajmera 8253 u svim MOD-ovima rada na sledeći način:

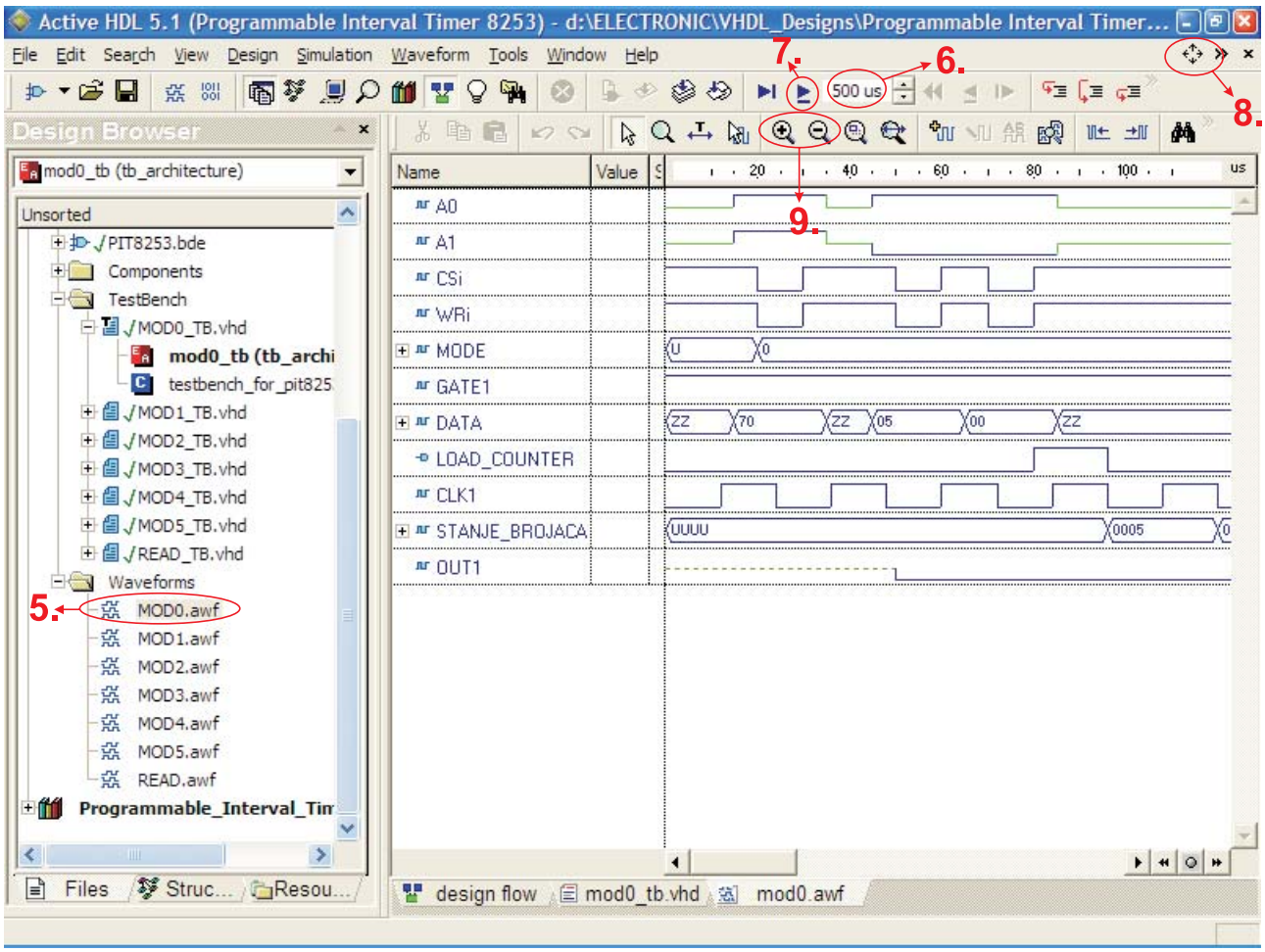
6.1 MOD 0:

1. Pokrenuti program *Active-HDL* čija se ikonica nalazi na *Desktop*-u
2. Otvoriti dizajn **Programmable Interval Timer 8253**
3. Otvoriti listing *TestBench*-a **MOD0_TB** koji se nalazi u *folder*-u *TestBench* **(**segment 1. na slici 24)
4. Upisati vrednost za brojanje koja je data u zadatku, umesto postojeće (segment 2. na slici 24)
5. Izvršiti kompajliranje pritiskom na uokvirenu ikonicu (segment 3. na slici 24)
6. Podesiti *Top level* entitet desnik klikom na ″+¨ ispred imena fajla **MOD0_TB.vhd** i izborom *Set as Top-Level* (segment 4. na slici 24)



Slika 24. Otvaranje dizajna i upisivanje početne vrednosti u brojač

7. Otvoriti *Waveform* fajl **MOD0.avf** koji se nalazi u folderu *Waveforms* (segment 5. na slici 25)
8. Podesiti vreme simulacije na 500 µs (us) ili više ako je potrebno (segment 6. na slici 25)
9. Pokrenuti simulaciju pritiskom na uokvirenu ikonicu (segment 7. na slici 25)
10. Uvećati *Waveform* editor na ceo prozor pritiskom na označenu ikonicu (segment 8. na slici 25)
11. Po završetku simulacije može biti potrebno da se vremenski dijagram uveća ili umanji radi bolje preglednosti. Komande za uveličavanje i umanjivanje su uokvirene crvenom bojom (segment 9. na slici 25)



Slika 25. Pokretanje simulacije

12. Proveriti koliko je taktova CLK-a, nakon startovanja brojača, potrebno da se izlaz brojača promeni sa logičke nule na logičku jedinicu
13. Odštampati dobijeni vremenski dijagram

Grupa 1:

LSB podatak _____ *Binarni oblik* _____ *Dekadni oblik*

MSB podatak _____ *Binarni oblik* _____ *Dekadni oblik*

Grupa 2:

LSB podatak _____ *Binarni oblik* _____ *Dekadni oblik*

MSB podatak _____ *Binarni oblik* _____ *Dekadni oblik*

Grupa 3:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 4:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 5:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 6:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 7:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 8:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 9:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 10:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 11:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 12:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 13:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 14:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 15:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 16:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

6.2 MOD 1:

1. Pokrenuti program *Active HDL* i učitati dizajn na isti način kao što je objašnjeno za **MOD 0**
2. Otvoriti listing *TestBench*-a **MOD1_TB** koji se nalazi u *folder*-u *TestBench*
3. Uneti zadatu vrednost umesto postojeće (u 94. redu kod-a **MOD1_TB.vhd**)
4. Izvršiti kompajliranje pritiskom na uokvirenu ikonicu (segment 3. na slici 24)
5. Podesiti *Top level* entitet desnik klikom na ″+¨ ispred imena fajla **MOD1_TB.vhd** i izborom *Set as Top-Level* (segment 4. na slici 24)
6. Otvoriti *Waveform* fajl **MOD1.avf** iz foldera *Waveforms* (segment 5. na slici 25)
7. Podesiti vreme simulacije na 500 μs (us) ili više ako je potrebno (segment 6. na slici 25)
8. Pokrenuti simulaciju pritiskom na uokvirenu ikonicu (segment 7. na slici 25)
9. Uvećati *Waveform* editor pritiskom na označenu ikonicu (segment 8. na slici 25)
10. Po završetku simulacije može biti potrebno da se vremenski dijagram uveća ili umanji radi bolje preglednosti. Komande za uveličavanje i umanjivanje su uokvirene crvenom bojom (segment 9. na slici 25)
11. Proveriti koliko je taktova CLK-a, nakon startovanja brojača, potrebno da se izlaz brojača promeni sa logičke nule na logičku jedinicu
12. Odštampati dobijeni vremenski dijagram

Grupa 1:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 2:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 3:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 4:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 5:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 6:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 7:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 8:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 9:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 10:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 11:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 12:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 13:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 14:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 15:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 16:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

6.3 MOD 2:

1. Pokrenuti program *Active HDL* i učitati dizajn na isti način kao što je objašnjeno za **MOD 0**
2. Otvoriti listing *TestBench*-a **MOD2_TB** koji se nalazi u *folder*-u *TestBench*
3. Uneti zadatu vrednost umesto postojeće (u 94. redu kod-a **MOD2_TB.vhd**)
4. Izvršiti kompajliranje pritiskom na uokvirenu ikonicu (segment 3. na slici 24)
5. Podesiti *Top level* entitet desnik klikom na ″+¨ ispred imena fajla **MOD2_TB.vhd** i izborom *Set as Top-Level* (segment 4. na slici 24)
6. Otvoriti *Waveform* fajl **MOD2.avf** iz foldera *Waveforms* (segment 5. na slici 25)
7. Podesiti vreme simulacije na 500 μs (us) ili više ako je potrebno (segment 6. na slici 25)
8. Pokrenuti simulaciju pritiskom na uokvirenu ikonicu (segment 7. na slici 25)
9. Uvećati *Waveform* editor pritiskom na označenu ikonicu (segment 8. na slici 25)
10. Po završetku simulacije može biti potrebno da se vremenski dijagram uveća ili umanji radi bolje preglednosti. Komande za uveličavanje i umanjivanje su uokvirene crvenom bojom (segment 9. na slici 25)
11. Proveriti koliko je taktova CLK-a potrebno da izlaz brojača pređe u stanje logičke nule
12. Odštampati dobijeni vremenski dijagram

Grupa 1:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 2:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 3:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 4:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 5:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 6:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 7:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 8:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 9:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 10:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 11:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 12:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 13:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 14:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 15:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 16:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

6.4 MOD 3:

1. Pokrenuti program *Active HDL* i učitati dizajn na isti način kao što je objašnjeno za **MOD 0**
2. Otvoriti listing *TestBench*-a **MOD3_TB** koji se nalazi u *folder*-u *TestBench*
3. Uneti zadatu vrednost umesto postojeće (u 100. redu kod-a **MOD3_TB.vhd**)
4. Isključiti *trigger* (postaviti signal GATE1 na '1' u 107.redu -460 μs)
5. Podesiti *Top level* entitet desnik klikom na ″+¨ ispred imena fajla **MOD3_TB.vhd** i izborom *Set as Top-Level* (segment 4. na slici 24)
6. Otvoriti *Waveform* fajl **MOD3.avf** iz foldera *Waveforms* (segment 5. na slici 25)
7. Podesiti vreme simulacije na 500 μs (us) ili više ako je potrebno (segment 6. na slici 25)
8. Pokrenuti simulaciju pritiskom na uokvirenu ikonicu (segment 7. na slici 25)
9. Uvećati *Waveform* editor pritiskom na označenu ikonicu (segment 8. na slici 25)
10. Po završetku simulacije može biti potrebno da se vremenski dijagram uveća ili umanji radi bolje preglednosti. Komande za uveličavanje i umanjivanje su uokvirene crvenom bojom (segment 9. na slici 25)
11. Proveriti koliko je taktova CLK-a izlaz brojača u stanju logičke nule a koliko u stanju logičke jedinice
12. Odštampati dobijeni vremenski dijagram

Grupa 1:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 2:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 3:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 4:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 5:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 6:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 7:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 8:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 9:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 10:

LSB podatak _____ _____

Binarni oblik *Dekadni oblik*

Grupa 11:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 12:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 13:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 14:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 15:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 16:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

6.5 MOD 4:

1. Pokrenuti program *Active HDL* i učitati dizajn na isti način kao što je objašnjeno za **MOD 0**
2. Otvoriti listing *TestBench*-a **MOD4_TB** koji se nalazi u *folder*-u *TestBench*
3. Uneti zadatu vrednost umesto postojeće (u 105. i 106. redu kod-a **MOD4_TB.vhd**-265 μs i 285 μs respektivno)
4. Dozvoliti brojanje brojača postavljanjem signala GATE1 na '1' u 111. redu kod-a **MOD4_TB.vhd** u 350 μs.
5. Podesiti *Top level* entitet desnik klikom na ″+¨ ispred imena fajla **MOD4_TB.vhd** i izborom *Set as Top-Level* (segment 4. na slici 24)
6. Otvoriti *Waveform* fajl **MOD4.avf** iz foldera *Waveforms* (segment 5. na slici 25)
7. Podesiti vreme simulacije na 500 μs (us) ili više ako je potrebno (segment 6. na slici 25)
8. Pokrenuti simulaciju pritiskom na uokvirenu ikonicu (segment 7. na slici 25)
9. Uvećati *Waveform* editor pritiskom na označenu ikonicu (segment 8. na slici 25)
10. Po završetku simulacije može biti potrebno da se vremenski dijagram uveća ili umanji radi bolje preglednosti. Komande za uveličavanje i umanjivanje su uokvirene crvenom bojom (segment 9. na slici 25)
11. Proveriti koliko je taktova CLK-a potrebno da izlaz brojača pređe u stanje logičke nule
12. Odštampati dobijeni vremenski dijagram

Grupa 1:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 2:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 3:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 4:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 5:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 6:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 7:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 8:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 9:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 10:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 11:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 12:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 13:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 14:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 15:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 16:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

6.6 MOD 5:

1. Pokrenuti program *Active HDL* i učitati dizajn na isti način kao što je objašnjeno za **MOD 0**
2. Otvoriti listing *TestBench*-a **MOD5_TB** koji se nalazi u *folder*-u *TestBench*
3. Uneti zadatu vrednost umesto postojeće (u 105. i 106. redu kod-a **MOD5_TB.vhd**-265 μs i 285 μs respektivno)
13. Podesiti *Top level* entitet desnik klikom na ″+¨ ispred imena fajla **MOD5_TB.vhd** i izborom *Set as Top-Level* (segment 4. na slici 24)
14. Otvoriti *Waveform* fajl **MOD5.avf** iz foldera *Waveforms* (segment 5. na slici 25)
15. Podesiti vreme simulacije na 500 μs (us) ili više ako je potrebno (segment 6. na slici 25)
16. Pokrenuti simulaciju pritiskom na uokvirenu ikonicu (segment 7. na slici 25)
17. Uvećati *Waveform* editor pritiskom na označenu ikonicu (segment 8. na slici 25)
18. Po završetku simulacije može biti potrebno da se vremenski dijagram uveća ili umanji radi bolje preglednosti. Komande za uveličavanje i umanjivanje su uokvirene crvenom bojom (segment 9. na slici 25)
19. Proveriti koliko je taktova CLK-a potrebno da izlaz brojača pređe u stanje logičke nule
4. Odštampati dobijeni vremenski dijagram

Grupa 1:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 2:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 3:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 4:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 5:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 6:

LSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

MSB podatak _____ (*Binarni oblik*) _____ (*Dekadni oblik*)

Grupa 7:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 8:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 9:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 10:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 11:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 12:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 13:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 14:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 15:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Grupa 16:

LSB podatak _____ _____
Binarni oblik *Dekadni oblik*

MSB podatak _____ _____
Binarni oblik *Dekadni oblik*

Literatura:

1. INTEL 8253 Programmable Interval Timer datasheet , November 1986.

2. INTEL 8254 Programmable Interval Timer datasheet , September 1993.

3. Mile K. Stojčev, Branislav D. Petrović ARHITEKTURE I PROGRAMIRANJE μRAČUNARSKIH SISTEMA ZASNOVANIH NA FAMILIJI PROCESORA 80x86 I izdanje,Elektronski fakultet, Niš, 1999.