Univezitet u Nišu
Elektronski fakultet
LEDA - *Laboratory for Electronic Design Automation*

Diplomski rad

Primena *ASIC Design Kit* alata za projektovanje integrisanih kola metodom standardnih ćelija

Zadatak:

- **Proučiti primenu ASIC Design Kit (ADK) alata za projektovanje integrisanih kola specifične namene zasnovanih na metodu standardnih ćelija**
- **Primeniti ADK za projektovanje integrisanog kola koje obavlja funkciju kalendara. Kolo se pobuđuje taktnim signalom frekvencije 32768Hz. Na izlazu generiše signal o trenutnom datumu (dan, mesec, godina = DD, MM, GGGG) u binarnom obliku. Omogućiti postavljanje početnog datuma. Ugraditi logiku za testiranje.**

Komisija:

1. **Prof. Dr. Predrag Petković**

   _____

2. **Prof. Dr. Vančo Litovski**

   _____

3. **Mr. Miljana Sokolović**

   _____

Kandidat:
Vladimir Petrović
datum prijavljivanja rada:

_____
datum predaje rada

_____
datum odbrane rada

_____

Zahvaljujem se svim članovima LEDA laboratorije koji su mi pomogli prilikom izrade diplomskog rada, a među njima bih posebno izdvojio Miljanu Sokolović, Marka Dimitrijevića, Miljana Nikolića i Bojana Anđelkovića. Posebno bih se zahvalio Prof. Dr Vanču Litovskom i svom mentoru Prof. Dr Predragu Petkoviću, na višegodišnjem strpljenju i naporu koji su uložili tokom mojih osnovnih studija, u cilju sticanja kvalitetnog znanja

1. Uvod

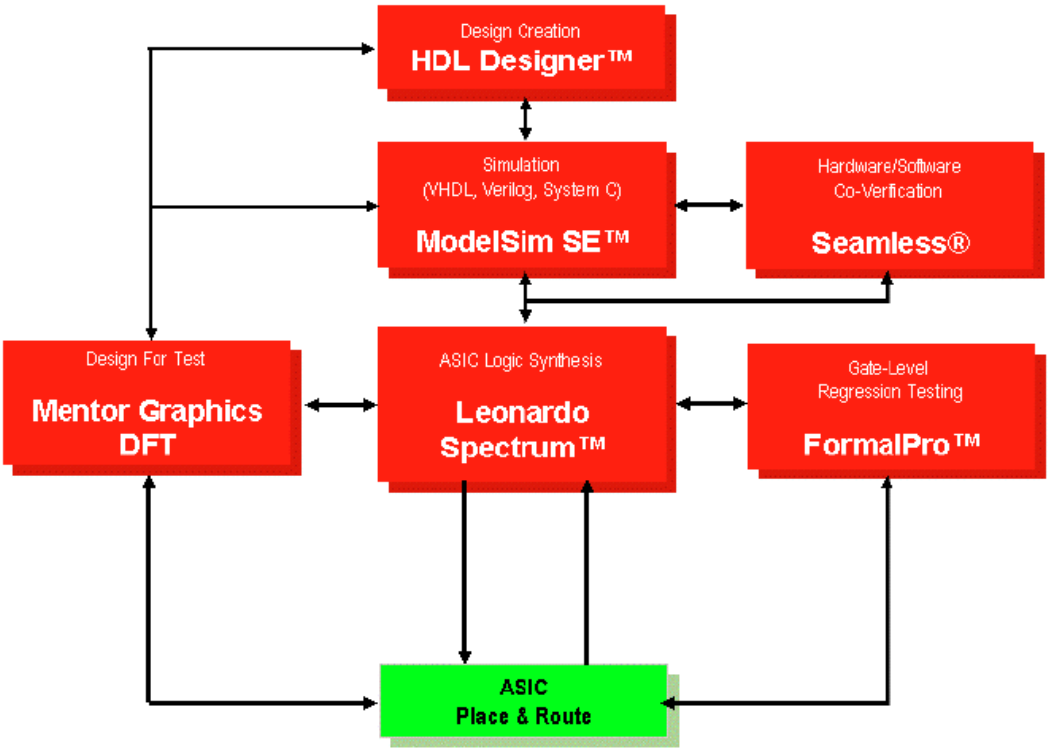
ASIC (*Application Specific Integrated Circuit*), što bi u prevodu značilo integrisana kola specifične namene, proizvode se isključivo za određenog korisnika. Nakon projektovanja *layout*-a kola, proizvodjaču se šalju podaci neophodni za izradu maski i nakon proizvodnje integrisanog kola, ono se isporučuje korisniku. Da bi korisnik mogao da projektuje kola specifične namene, on od vlasnika tehnologije u kojoj će kolo biti proizvedeno, dobija sve neophodne tehnološke podatke. [Pet05]

Prilikom projektovanja integrisanog kola (ASIC) koristi se skup logičkih modula. U okviru svake tehnologije specifirane su I/O karakteristike modula. Moduli su lokalno projektovani, tako da je poznat redosled i položaj portova preko kojih se moduli mogu povezivati. Proizvodjač integrisanog kola vrši gore pomenuto lokalno projektovanje modula. Svi podaci potrebni za dalje projektovanje, smeštaju se u biblioteke modula [Pet05].

Projektovanje bazirano na principu korišćenja modula može se razložiti na:

- prevodjenje funkcionalnog opisa u strukturu koja sadrži module iz biblioteka
- rasporedjivanje modula po ploči (*placement*)
- povezivanje modula (*routing*)
- prevodjenje *layout*-a u format pogodan za izradu maski

U ovom radu biće objašnjen postupak projektovanja integrisanog kola pomoću *ASIC Design Kit* (ADK) alata koji distribuira *Mentor Graphics*. ADK obuhvata alate za logičku sintezu, planiranje površine, razmeštaja ćelija i povezivanje. Na slici 1. dat je šematski prikaz kako od VHDL opisa projekta doći do *layout*-a integrisanog kola uz pomoć gore pomenutih alata.



Slika 1. Blokovski prikaz kretanja ASIC dizajna

Opis primene alata biće ilustrovan na projektovanju integrisanog kola koje obavlja funkciju kalendara. Pri tome, koristiće se metod projektovanja zasnovan na primeni standardnih ćelija. Dakle, prilikom projektovanja integrisanog kola, najpre je potrebno adekvatno opisati module na nekom od jezika za opis hardvera (HDL). U ovom slučaju korišćeni su VHDL opisi, koji su dati u prilogu A. U cilju kompajliranja i simuliranja opisanih modula kao i kompletnog kola, korišćen je *Model Sim SE* alat. Detaljniji opis i način korišćenja ovog programa dat je u drugom poglavlju.

Nakon uspešnog kompajliranja i simulacije VHDL kodova, potrebno je izvršiti sintezu. Alat koji se koristi u te svrhe jeste *Leonardo Spectrum*. Rezultat sinteze dobija se u obliku dva fajla, jedan koji omogućava verifikaciju rada sinetizovanog kola i drugi koji će se, kasnije, koristiti u alatima za razmeštanje i povezivanje ćelija – *Place and routing*. Fajl namenjen verifikaciji nakon sinteze može da se generiše u više izlaznih formata, kao što su Verilog, VHDL, eddif itd. Način korišćenja pomenutog alata za sintezu dat je u trećem poglavlju.

Priprema za generisanje *Layout*-a sprovodi se kroz dva softverska alata. Nakon sinteze modula, kako je gore navedeno, dobijamo dva fajla od kojih nam je jedan potreban za izradu *layout*-a. U pitanju je netlista u Verilog formatu. Alat *Design Architect* importuje Verilog netlistu i priprema je za dalju obradu. Način korišćenja ovog alata dat je u četvrtom poglavlju. Nakon izvršenih potrebnih akcija prelazi se u *IC Station*, pomoću koga se generiše *layout*-a čipa. Opis ovog alata dat je u petom poglavlju.

U šestom poglavlju opisan je projektni zadatak koji služi za ilustraciju primene opisanih alata. Radi se o ASIC kolu koje treba da obavlja funkciju kalendara. Očekuje se da kolo pobuđuje takt frekvencije 32768Hz i da nakon isteka odgovarajućeg vremena ažurira stanje na izlazu posle intervala od jednog dana. Izlazni signal dobija se u obliku cifara koje označavaju dan, mesec i godinu u BCD formatu kao DD.MM.GGGG. Treba omogućiti direktno postavljanje vrednosti na izlazu kalendara. Pored toga, neophodno je ugraditi logiku za testiranje hardvera.

U sedmom poglavlju dati su rezultati testiranja, kao i delovi kola na koje treba obratiti pažnju prilikom testiranja fabrikovanog kola.

Detalji realizacije projekta dati su u prilozima i to:

- prilog A prikazuje opise pojedinih modula projekta primenom VHDL jeziku za opis hardvera.
- prilog B daje logičke šeme sintetizovanih modula;
- prilog C sadrži lejaut svakog modula
- prilog D prikazuje *layout* kompletnog čipa sa stopicama za napajanje i povezivanje sa spoljašnjim svetom.

NAPOMENA: U radu nisu data detaljna objašnjenja o svakom alatu jer se ona mogu naći u odgovarajućim priručnicima [Adk1, Adk2, Adk3 i Adk4]. Umesto toga, rad je koncipiran kao uputstvo za korišćenje pojedinih alata. Zato su sve opisane aktivnosti ilustrovane na jednostavnijim primerima. Posebno napominjemo da je ADK instaliran pod UNIX operativnim sistemom. Ovo implicira da su za praćenje opisa rada alata neophodna osnovna znanja iz te oblasti. Najčešća greška kojoj su skloni korisnici naviknuti na Windows OS odnosi se na razliku u značenju malih i velikih slova. Poželjno je da korisnik alata ima određenog iskustva sa HDL dizajnom.

2. ModelSim

ModelSim predstavlja program pomoću koga je moguće unositi, kompajlirati i simulirati opise logičkih funkcija digitalnih kola. Sastavni je deo paketa za projektovanje integrisanih kola firme *Mentor Graphics* [Adk1]. Uz standardne opcije koje poseduje, kao i svaki program ove namene, izdvaja se zbog mogućnosti optimizacije VHDL opisa za sintezu. Ovaj softverski alat, pomaže projektantu da u samom početku projektovanja uoči eventulane delove kôda koje nije moguće sintetizovati, što znatno utiče na uštedu vremena.

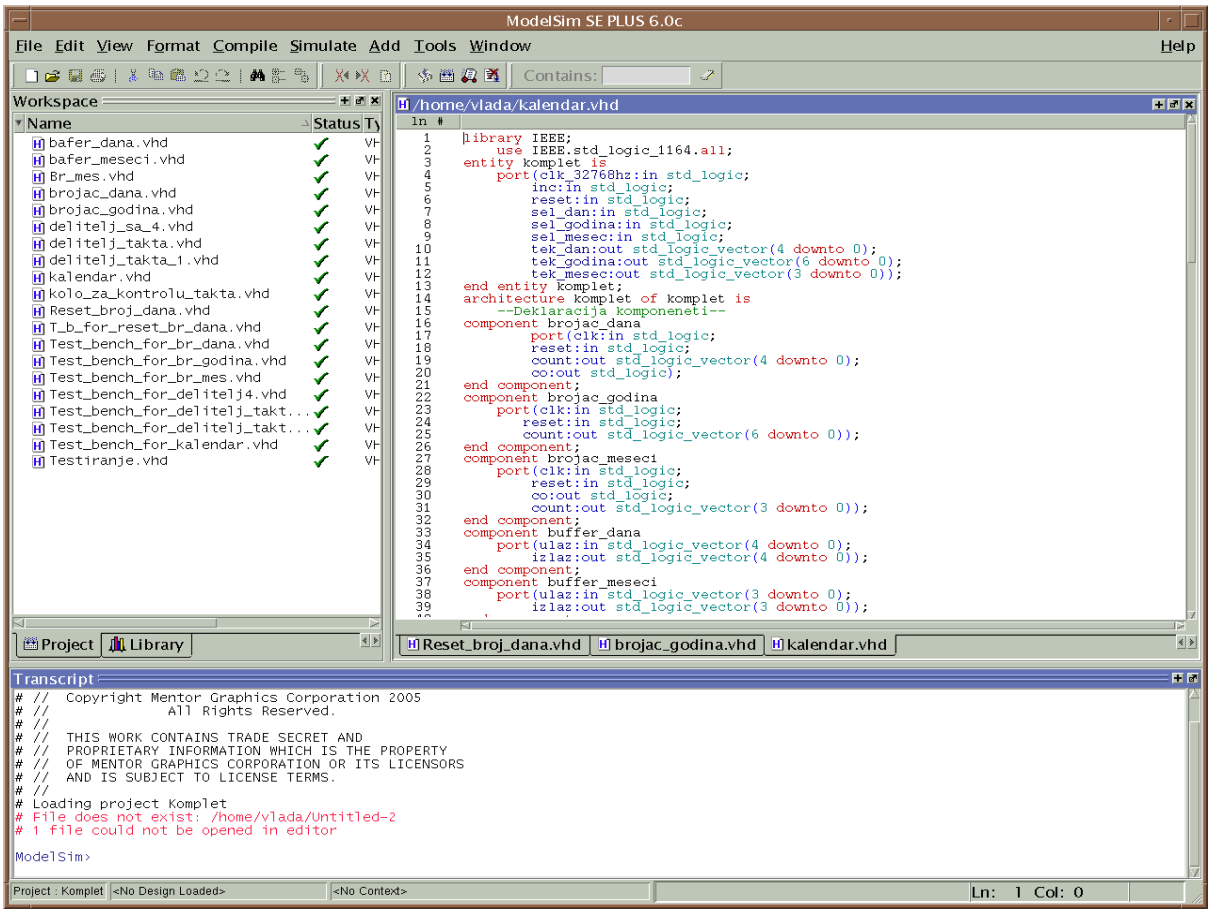
Kompajliranje, korak koji dolazi pre simuliranja, predstavlja proveru koda sa stanovišta grešaka. Greške koje se javljaju u opisu mogu biti sintaksne, greške vezane za pozivanje željenih biblioteka, kao i sve ostale greške koje se mogu javiti prilkom odstupanja od pravila koja su definisana jezikom za opis hardvera. Simulacija služi za proveru ispravnosti funkcionisanja prethodno kompajliranog opisa. Za simulaciju je potrebno generisanje test vektora. Korisno je u ovom delu napomenuti da je potrebno opisati *test bench* kodove za svaki modul. *Test bench* predstavlja kod koji se vezuje za entitet koji se testira, s tim što se generisanje test vektora definiše jedanput. Ovo je značajno kada se ima u vidu da je potrebno testirati neko kolo (verifikovati projekat) više puta posle različitih faza projektovanja čime se postižu značajne uštede vremena. [Adk1]

Pokretanje programa

Program se pokreće pozivom

%vsim

Na slici 2. prikazano je radno okruženje nakon aktiviranja programa.

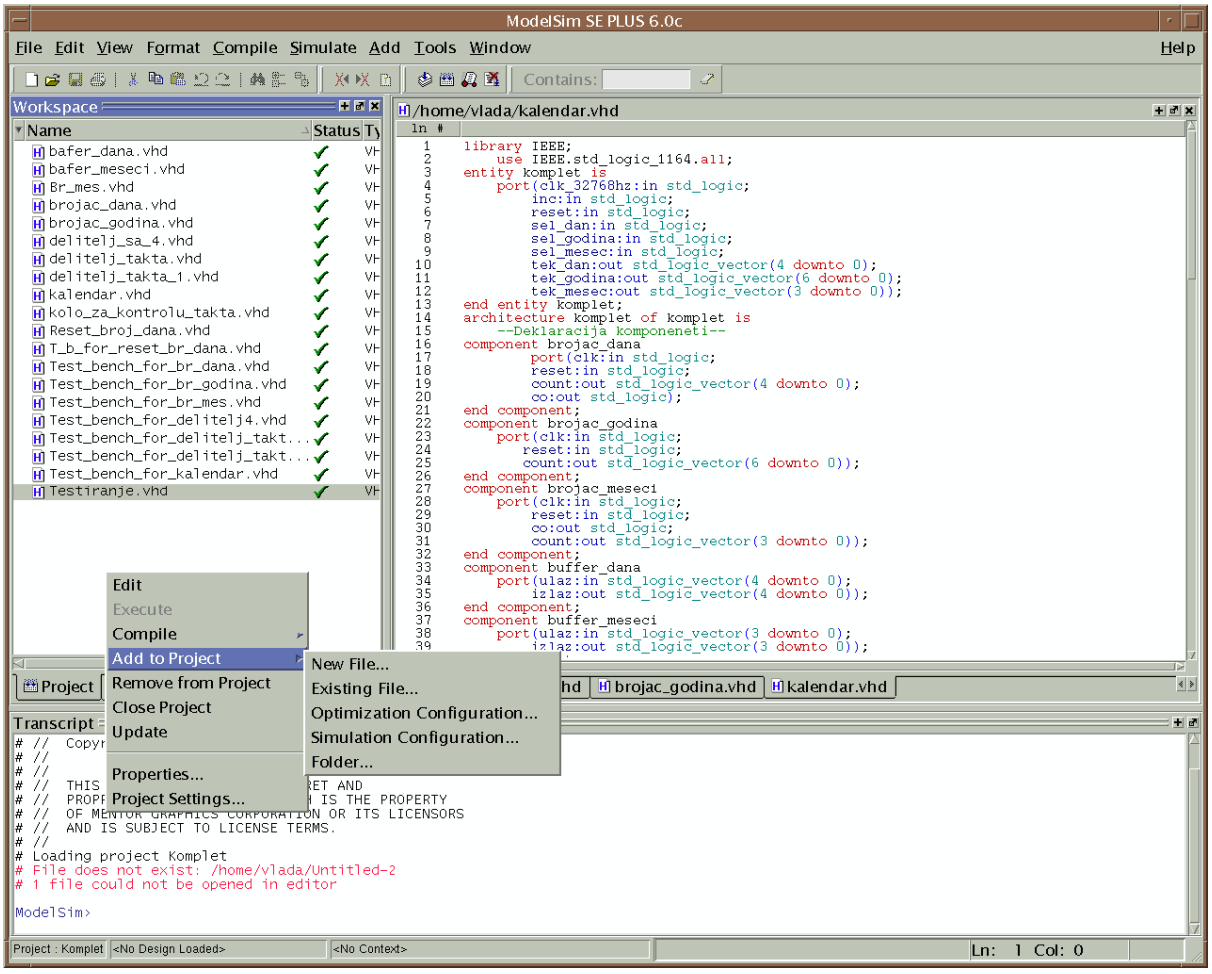


Slika 2. Radno okruženje programa *ModelSim*

Prozor *Workspace* sadrži listu opisa priključenih datom projektu. Isto tako i sve biblioteke koje se mogu videti klikom na tab *library*. Medju bibliotekema nalazi se i radna biblioteka *work* u koju se smeštaju kompajlirani opisi, koji će se kasnije koristiti u simulaciji. Duplim klikom na bilo koji od kodova datih u *wrkspace*-u, pojaviće se, sa desne strane, listing koda sa obeleženim redovima U primeru sa slike 2 prikazan je VHDL opis iz datoteke 'kalendar.vhd'. U donjem delu prozora (*Transcript*) nalazi se oblast, koja pored uloge da korisnika izveštava o stanju i eventualnim greškama u projektu (prilikom kompajliranja i simulacije), ima i ulogu zadavanja odredjenih parametara, što će u daljem delu teksta biti pokazano.

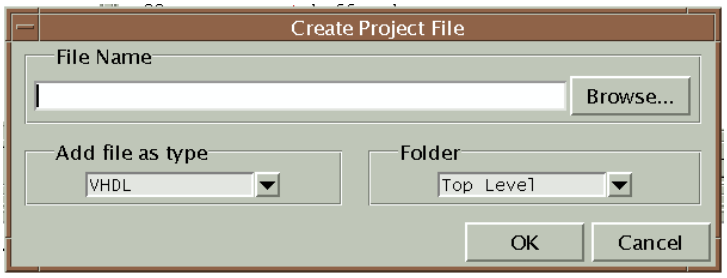
Kreiranje novog opisa

Ako se želi kreiranje novog opisa, potrebno je desnim klikom u oblasti *workspace* (tab *project*) selektovati *add to project -> new file* (slika 3).



Slika 3. Kreiranje novog opisa - korak 1

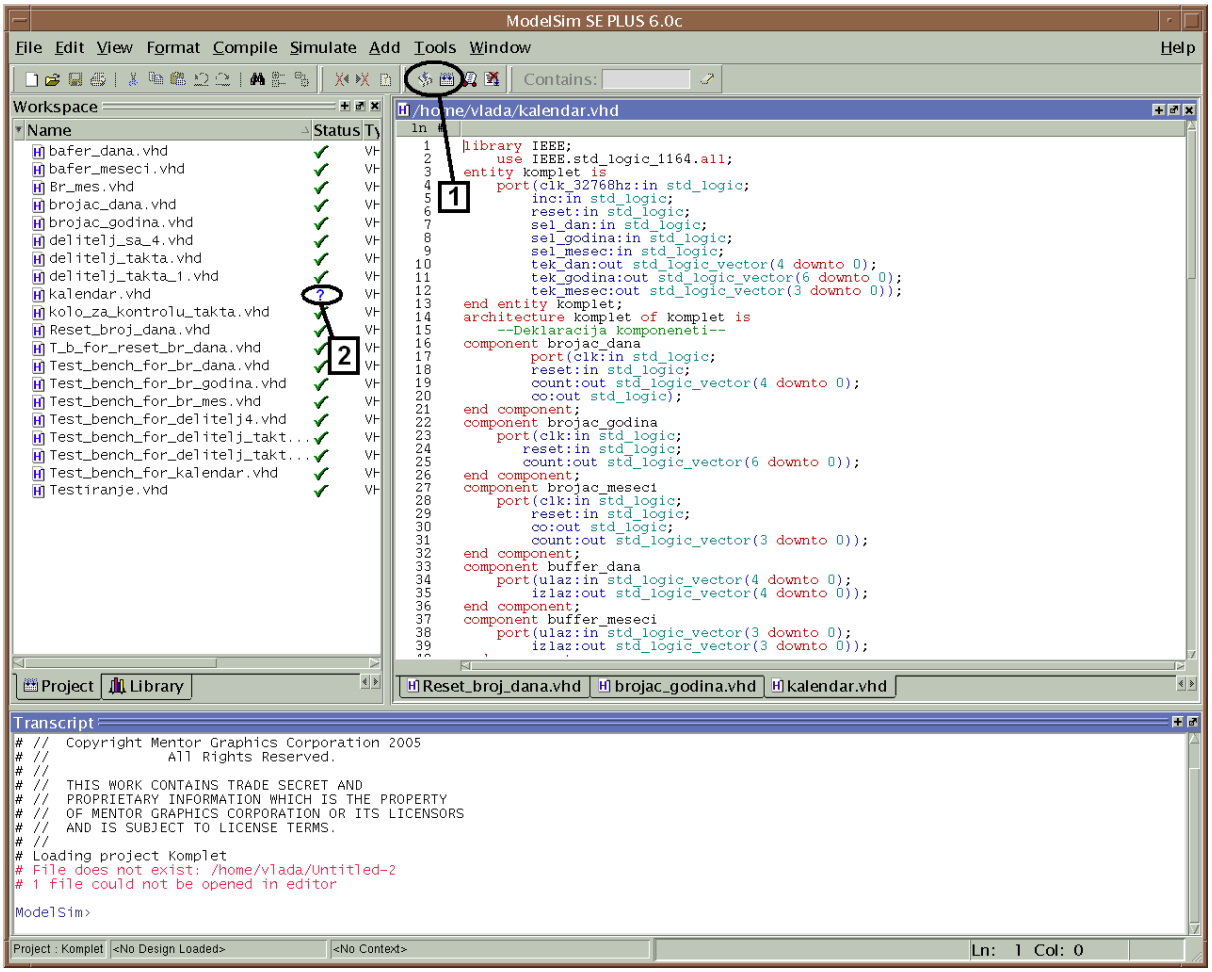
U *dialog box*-u, datom na slici 4, koji aktivira akcija *new file*, potrebno je upisati ime fajla koji kreiramo kao i izabrati njegov tip (VHDL, Verlog, System Verilog itd). Ako je u pitanju projekat sa više nivoa hijerarhije, moguće je pojedine nivoe hijerarhije razdvojiti izborom u *folder* opciji.



Slika 4. *Dialog box* za kreiranje novog opisa

Kompajliranje koda

Nakon ovih akcija, u *workspace*-u će biti postavljen nov blanko opis, a duplim klikom na njega, pojaviće se, sa desne strane, prazan listing u koji je potrebno uneti odgovarajući kod. U okviru *workspace*-a, u oblasti *status* stajaće znak '**?**' (na slici 5, polje obeleženo brojem 2). Ovaj znak govori o tome da je potrebno kôd kompajlirati. Kompajliranje se obavlja klikom na polje, na koje ukazuje broj 1, sa slike 5.



Slika 5. Kompajliranje koda

Oblast na koju ukazuje broj 1. sastoji se od dve ikonice, od kojih ikonica levo služi za kompajliranje selektovanog fajla, a ikonica desno za kompjliranje svih fajlova u projektu. U slučaju da postoje sintaksne greške, u *status* polju, posle kompajliranja, umesto **?**, stajaće znak ***X***. U slučaju da je kod sintaksno ispravan pojaviće se znak √.

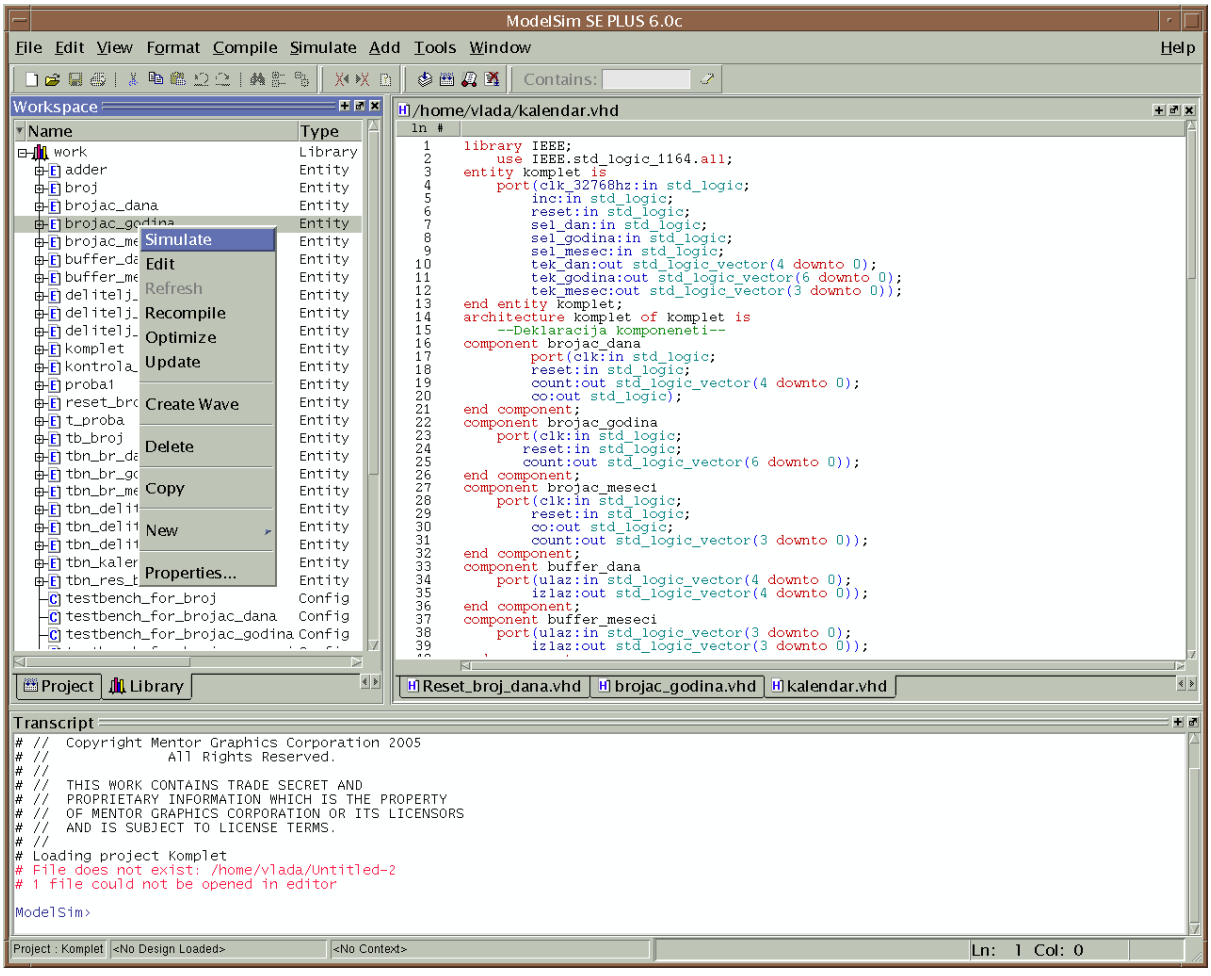
Prilikom projektovanja sistema koji se sastoje iz više komponenti, poželjno je nakon uspešnog kompajliranja izvršiti simulaciju svake od komponenti. Time se štedi vreme u delu

projekta u kome je potrebno da se sve komponente povežu i simulira ceo projekat. U slučaju da se simulacija obavi na kraju, kada su sve komponente povezane, teže je uočiti koja komponenta ne funkcioniše onako kako je specifikacijom projekta predviđeno.

Nakon verifikacije funkcionalnih opisa svih pojedinih blokova (koji čine sistem), pristupa se opisu, kompajliranju i simuliranju opisa na najvišem nivou - *top-level* opisa, koji objedinjuje sve komponente projekta. Rezultati simuliranja ovog opisa, predstavljaju idealizovanu sliku rada kalendara. Realniji rezultati simulacije, dobiće se nakon verifikacije sintetizovanog kola kalendara.

Simulairanje opisa

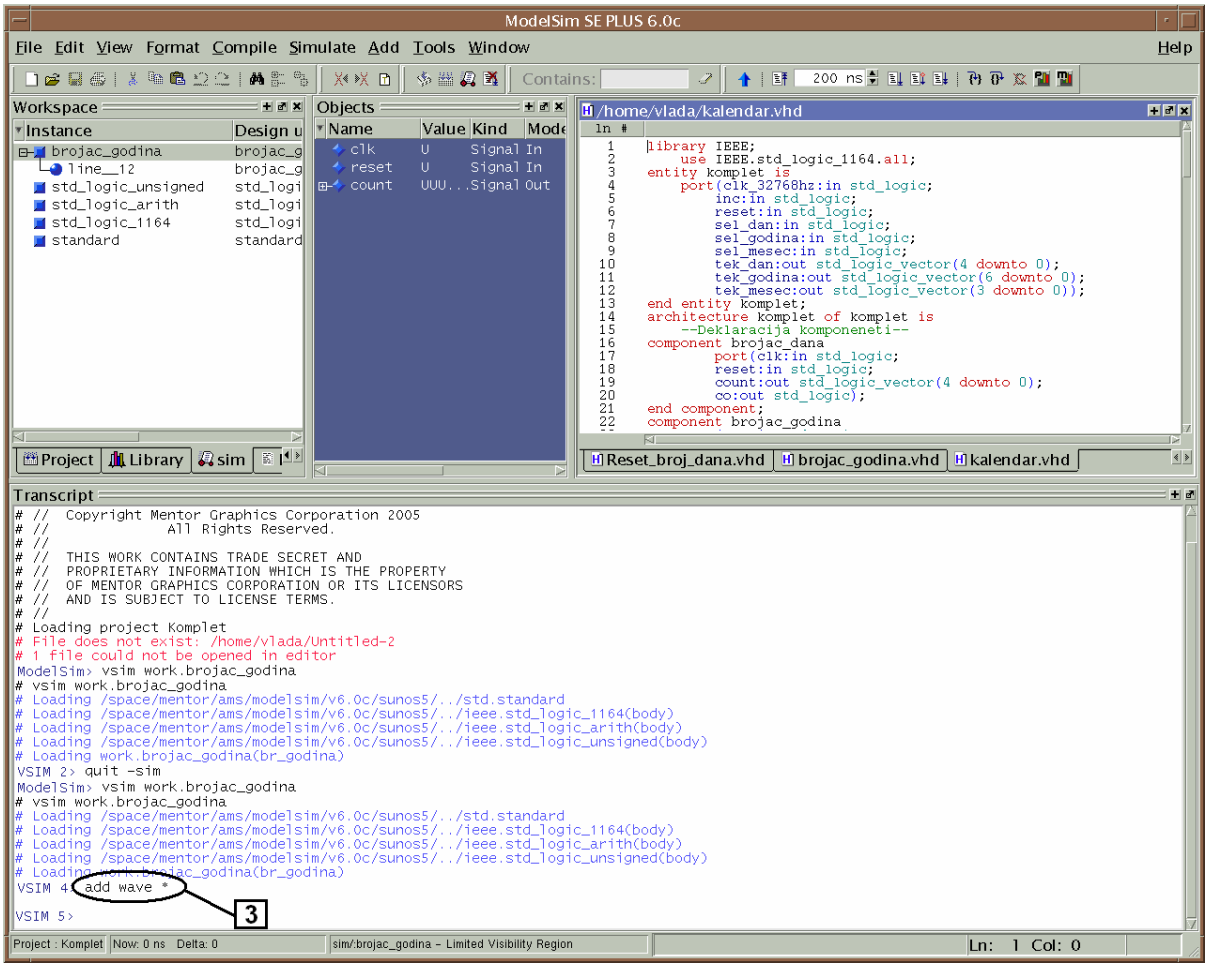
Simulacija se pokreće tako što se, najpre, u *workspace*-u prebacimo na tab *library*. U okviru ovog tab-a, potrebno je ući u direktorijum *work*, gde se nalaze iskompajlirani kodovi. Selekcijom neke od komponenti, gde su komponente date imenom entiteta, i klikom na desni taster miša, pojaviće se meni dat na slici 6. Na kome je potrebno kursor miša dovesti do opcije *Simulate* i inicijalizirati simulaciju.



Slika 6. Inicijalizacija simulacije opisa *brojac_godina*

Na narednoj slici 7. prikazano je okruženje nakon inicijalizacije simulacije datog opisa.

Da bi se nastavilo sa simulacijom, potrebno je u *transcript*-u upisati naredbu za pozivanje *wave* dijagrama – *add wave* * (oblast označena brojem 3 na slici 7).



Slika 7. Pozivanje *wave* dijagrama

Pritiskom na <enter>, pojaviće se u gornjem desnom uglu *wave* dijagram prikazan na slici 8.

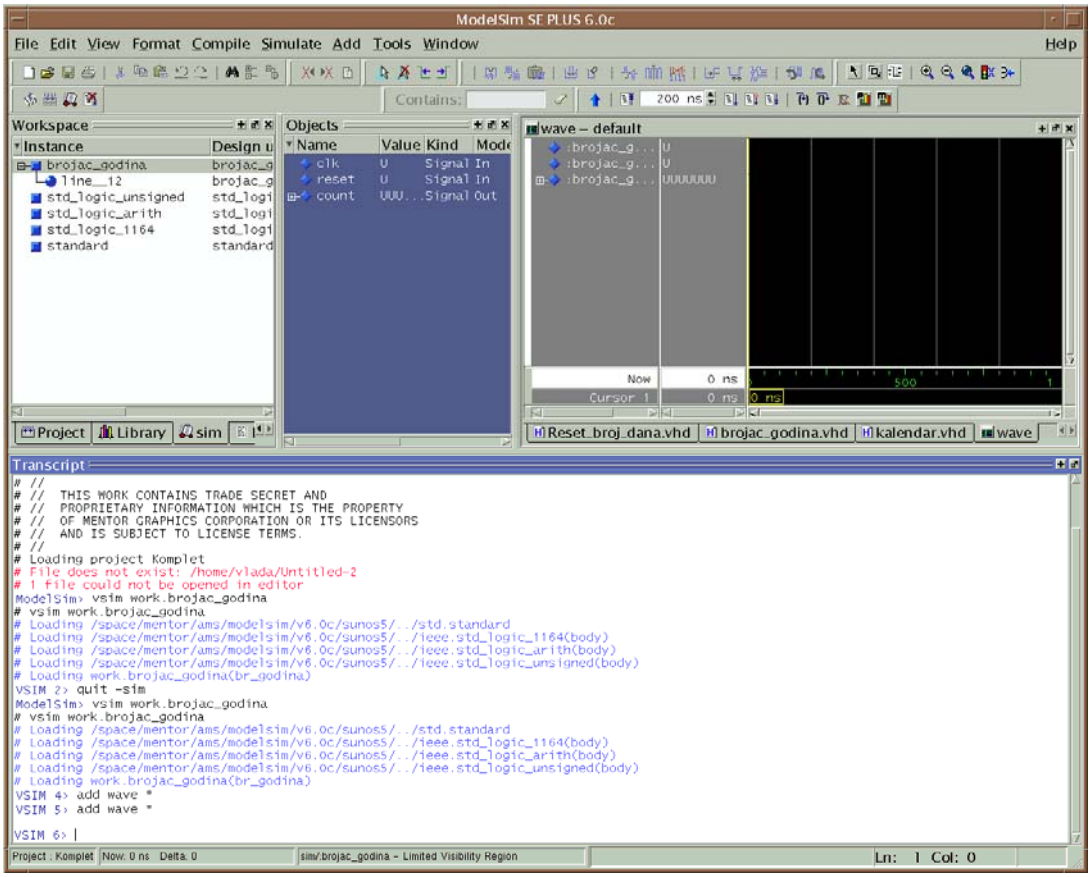
Nakon aktiviranja *wave* dijagrama potrebno je definisati signale pobude. Signali pobude se definišu na dva načina. Prvi način je korišćenjem naredbe *force* u okviru *transcript* prozora. Ovaj način daje dosta mogućnosti za definisanja signala, kao na primer kada je potrebno pre početka rada sistem resetovati itd. Drugi način jeste korišćenje prečica koje se aktiviraju kao padajući meniji. Dovoljno je samo selektovati ulazni signal u *wave* prozoru i desnim klikom aktivirati padajući meni. U okviru padajućeg menija, sa slike 9 označenog brojem 4, moguće je uočiti postavljanje signala na odredjenu vrednost, ukidanje signala, kao i postavljanje takt signala.

Preporuka proizvođača je da se koristi *transcript* zadavanje, jer je zadavanjem preko padajućih menija lakše načiniti grešku (u SE verziji mnoge naredbe ponuđene u padajućim menijima se ne mogu izvršiti. [Adk1] Tako, na primer, ako želimo da zadamo signal takta ***clk***, periode 10 ns sa početnom vrednošću 0 i jedan reset signal u trajanju od 4 ns, u okviru *transcript* prozora upisaćemo:

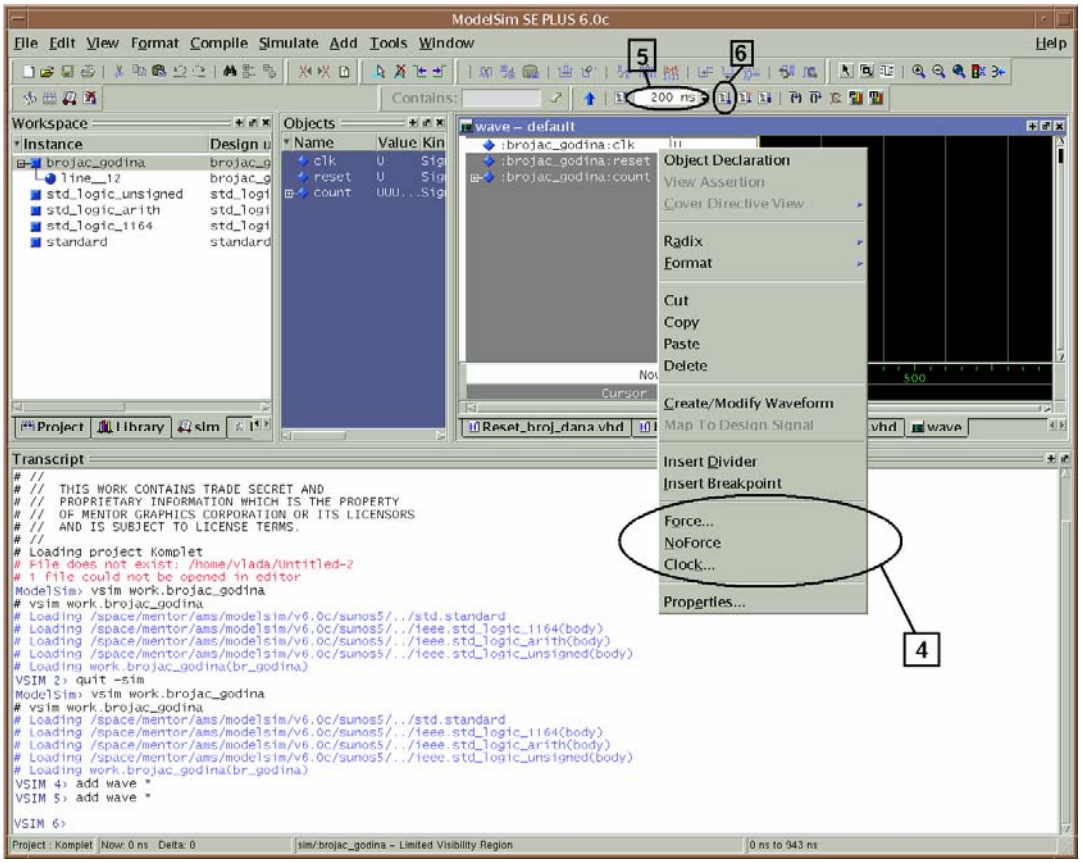
> force clk 0 0 ns, 1 {5 ns} –r 10 ns

> force reset 1 0 ns, 0 4 ns

Napomena: > je oznaka odzivnog znaka (prompt) *trancript* editora. Potrebno je nakon svake *force* naredbe pritisnuti taster <*enter*>, kako bi bilo moguće unošenje nove pobude.

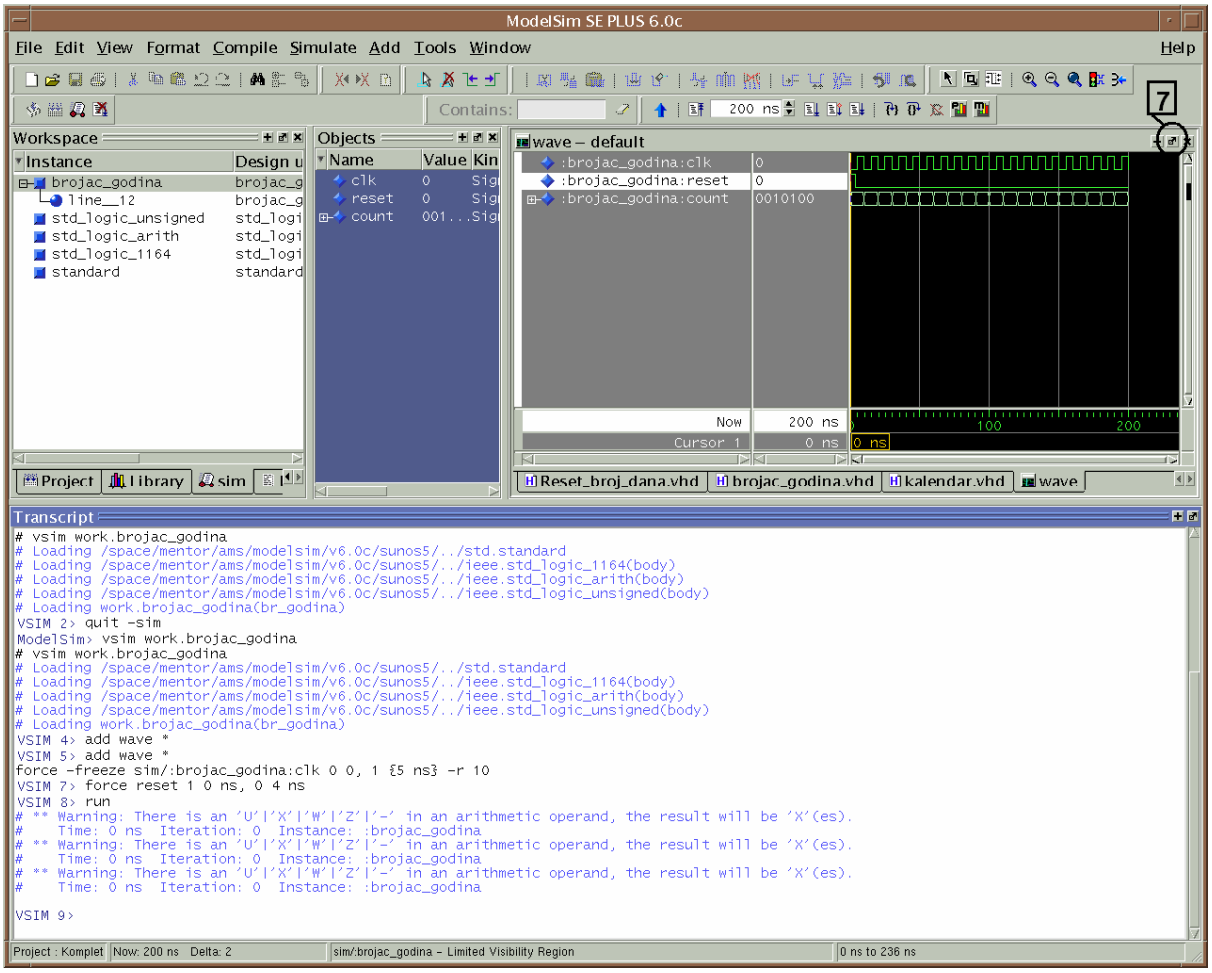


Slika 8. Aktiviranje *wave* dijagrama



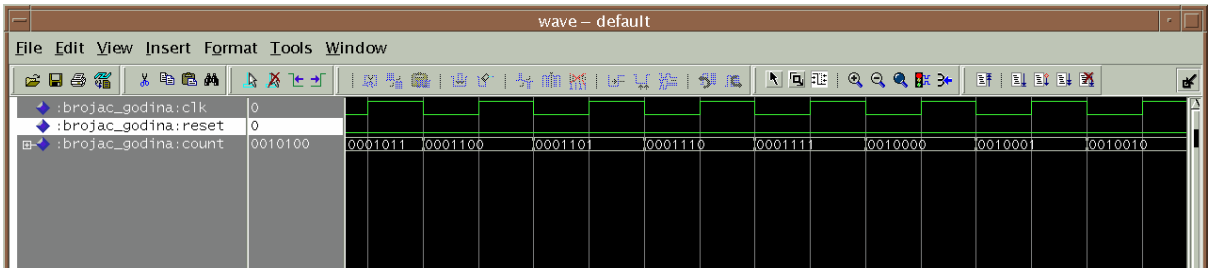
Slika 9. Zadavanje parametara simulacije

Nakon zadavanja svih pobudnih signala završena je inicijalizacija simulacije. Sledeći korak je definisanje trajanja simulacije (polje broj 5, slika 9) i pokretanje simulacije (polje na koje ukazuje broj 6, slika 9). Nakon simulacije, u okviru *wave*-a, pojaviće se spisak signala koji odgovaraju opisu projektovanog kola. Promena načina predstavljanja signala (*unsigned, hexadecimal, binary,* itd) može se vršiti odabiranjem *Radix* opcije sa padajućeg menija selektovanog signala. Na slici 10. dati su rezultati simulacije opisa 'brojača_godina'.



Slika 10. Prikaz rezultata simulacije

Ako se želi precizniji prikaz rezultata, klikom na polje koje označava broj 7 (*undock*), rezultati simulacije t.j. *wave* dijagram, biće izdvojen kao zaseban prozor (slika 11). Za detaljnije informacije korisnik se upućuje na [Adk2].



Slika 11. *wave* kao zaseban prozor

Sledeći korak u projektovanju jeste sinteza kola, kojom se funkcionalni opis transformiše u strukturni.U tu svrhu, u okviru ADK paketa koristi se alat za sintezu *Leonardo Spectrum*.

3. Leonardo Spectrum

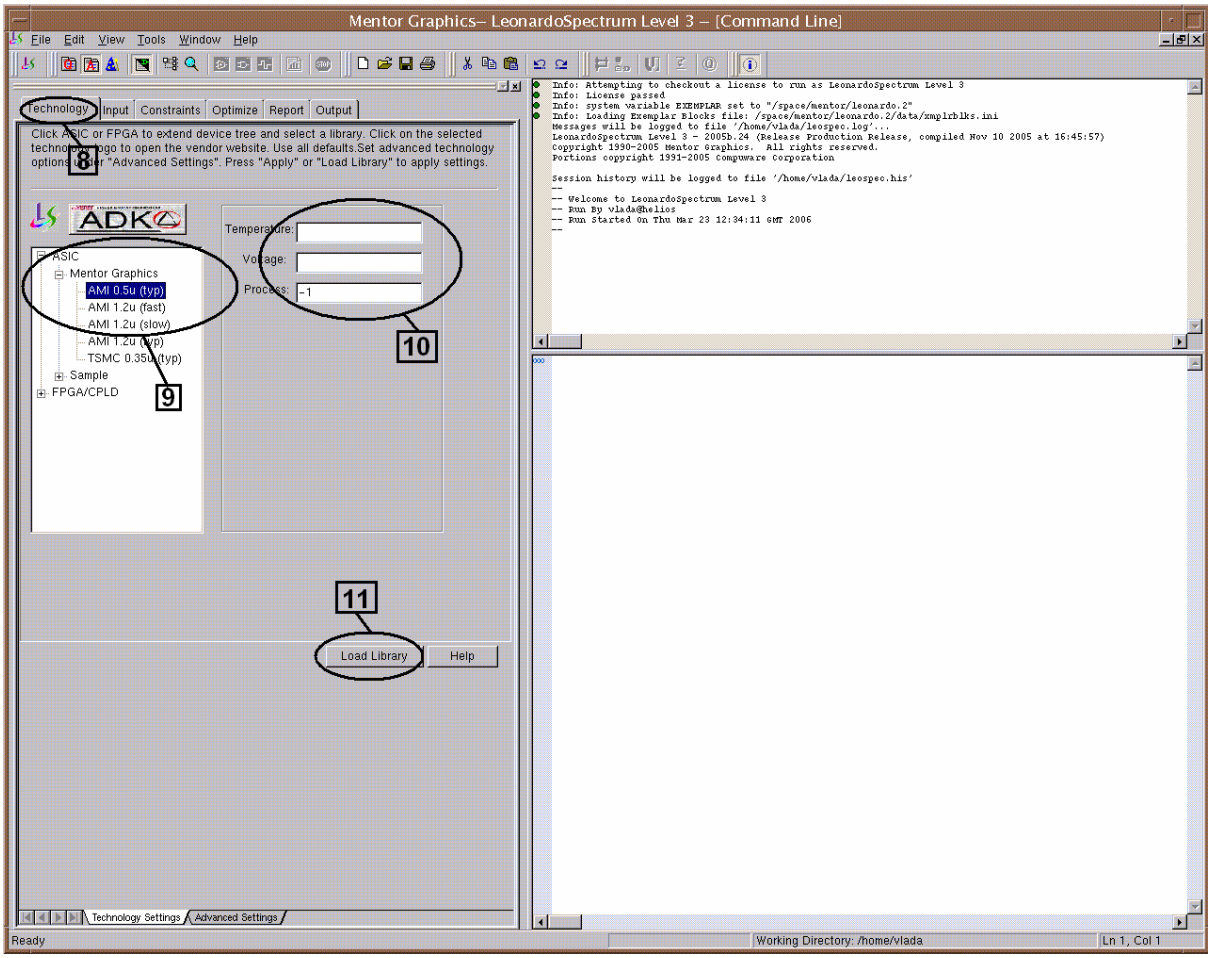
Alat za sintezu kola, uprošćeno govoreći, omogućuje prevodjenje HDL opisa u hardver. Svojim algoritmima, alat za sintezu prepoznaje karakteristične delove koda i na osnovu biblioteke t.j. tehnologije za koju se kolo projektuje, povezuje module koji su u datoj tehnologiji dostupni. Zato je i potrebno, prilikom opisa kola, voditi računa o pravilima projektovanja za sintezu. *Leonardo Spectrum* je veoma jednostavan za korišćenje. U narednom delu teksta biće prikazano kako koristi program za sintezu HDL koda.

Pozivanje programa

Program se pokreće pozivom:

% leonardo, gde je % *prompt* signal (javljaće se i u daljem delu teksta).

Pojaviće se *dialog box* za potvrdu licence, gde je porebno selektovati treću opciju (*Leonardo Spectrum Level 3*), ako već nije automatski selektovana i kliknuti na OK. Pojaviće se prozor prikazan na slici 12.



Slika 12. Početni prozor za učitavanje biblioteke odgovarajuće tehnologije

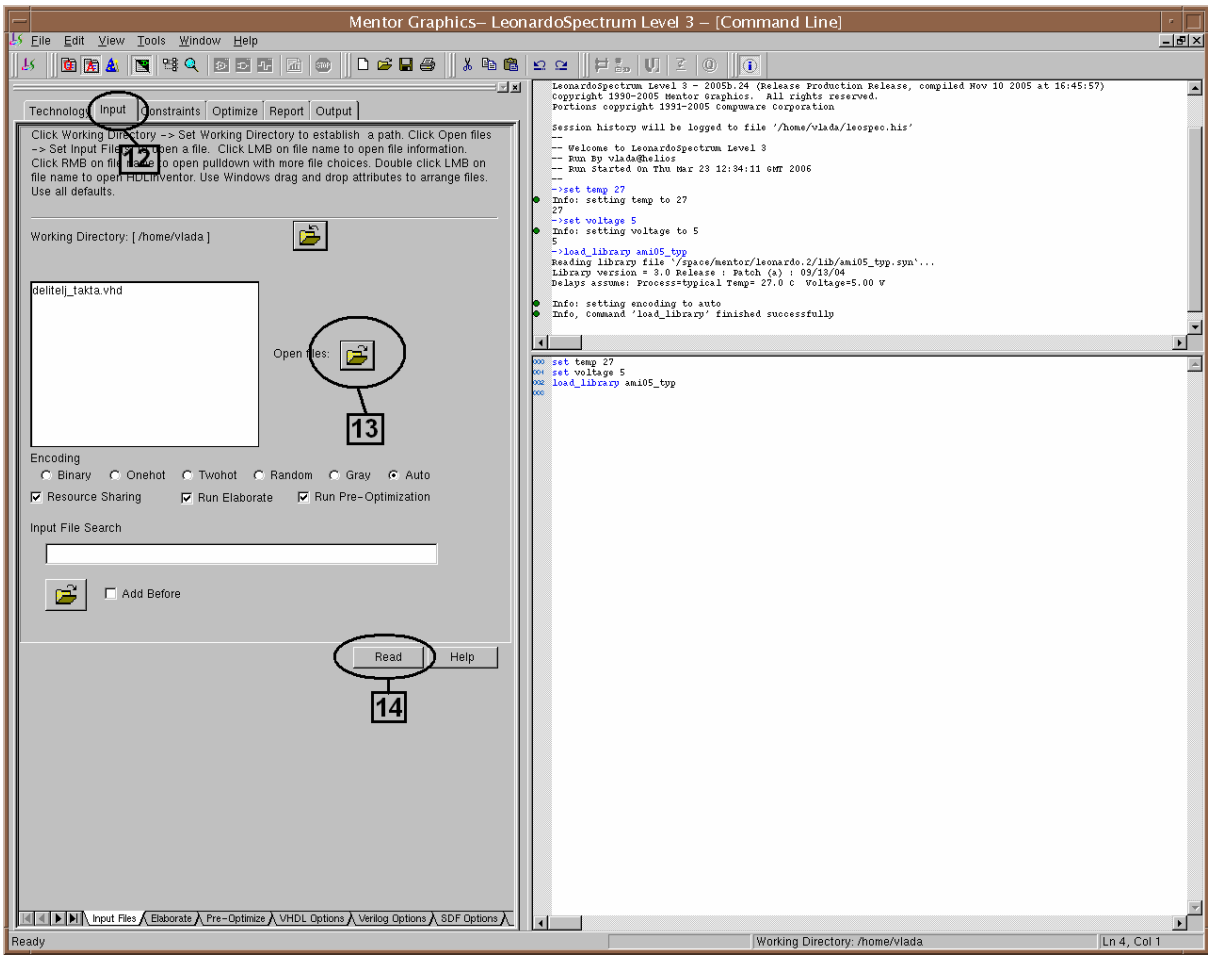
Učitavanje biblioteke željene tehnologije

U zavisnosti od toga da li se želi projektovati ASIC ili FPGA čip, bira se i lista mogućih tehnologija, na slici 12. obeležena brojem 9. Kako je cilj ovog rada projektovanje ASIC čipa, potrebno je selektovati neku od ASIC biblioteka. U radu sa studentskom verzijom paketa za projektovanje koristi se *ami05(typ)* tehnologija. U oblasti označenoj brojem 10, potrebno je uneti temperaturu (oko 27 ºC) i odgovarajući napon napajanja kola koji će biti primenjen (obično 5V).

Nakon svih gore navedenih koraka, potrebno je učitati biblioteku klikom na polje označeno brojem 11. U gornjem desnom delu prozora program će prijaviti uspešno ili neuspešno učitanu biblioteku.

Učitavanje fajlova za sintezu

Prelaskom na tab *input* (oblast broj 12 na slici 13.), učitava se HDL opis što predstavlja sledeći korak u inicijalizaciji sinteze.



Slika 13. Učitavanje HDL opisa

Učitavanje fajlova za sintezu obavlja se pritiskom na dugme označeno brojem 13. Selektovani fajl se ubacuje u listu fajlova za sintezu. Opcija kodiranja stanja služi da se definiše način kodiranja stanja ako projekat koji se sintetizuje sadrži koonačni automat. Opciono se postavlja na automatski način kodiranja. Nakon svih potrebnih postavki potrebno je učitati fajl, pritiskom na *read*, polje označeno brojem 14.

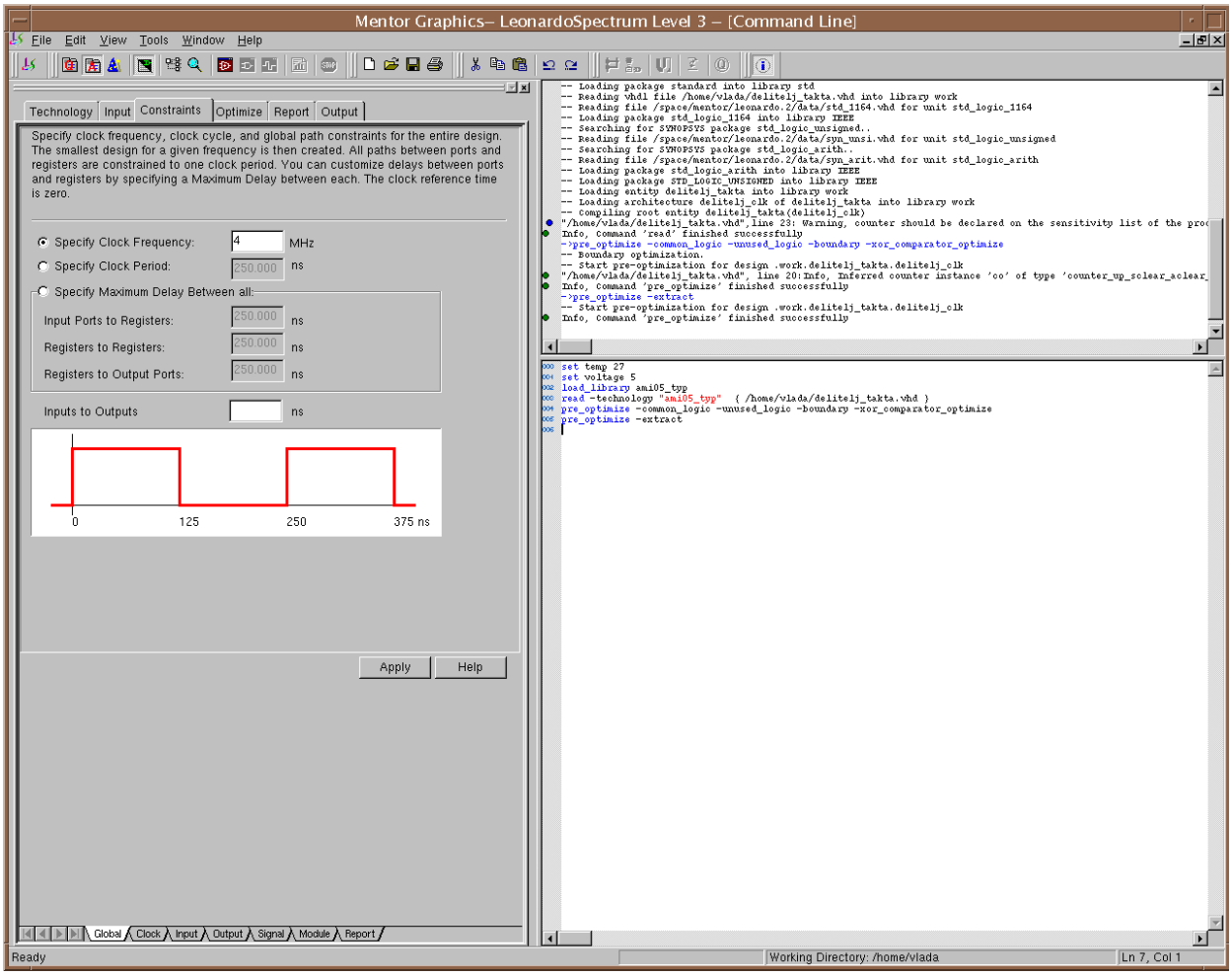
Definisanje vremenskih parametara

Sledeći korak jeste definisanje vremenskih ograničenja u kolu selekcijom tab-a *constraints*. Moguće je definisati sledeće kriterijume po kojima se obavlja sinteza:

- frekvencija takta
- vremena kašnjenja i to
  - izmedju portova i registara,
  - izmedju samih registara kao i

- izmedju registara i izlaznih portova.

Nakon definisanja ovih parametara potrebno je kliknuti na *apply*, čime će biti izvršena pre-optimizacija. Okruženje vezano za ovaj korak prikazano je na slici 14.



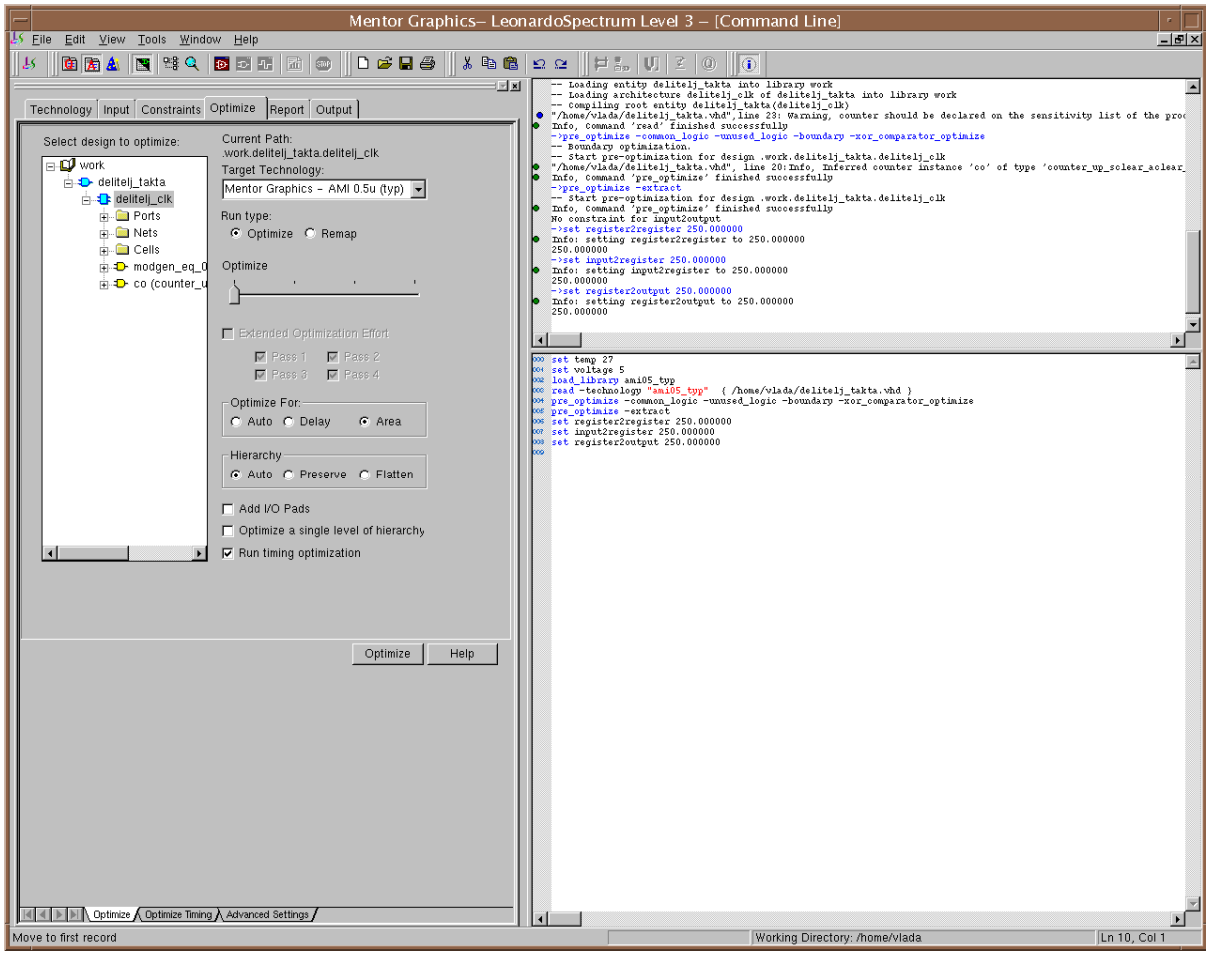
Slika 14. Postavljanje vremenskih parametara sinteze

Optimizacija

Selekcijom tab-a *optimize*, prelazi se u naredni korak, gde je potrebno postaviti parametre prema slici 15.

NAPOMENA: Potrebno je, prilikom sinteze na najvišem nivou opisa, *top-level*, da se čekira opcija za dodavanje stopica, *add I/O pads*.

U okviru tab-a *report* definiše se ime izlaznog tekst fajla vezanog za rezultate sinteze. Potrebno je navesti ime fajla i pritisnuti *report area*.

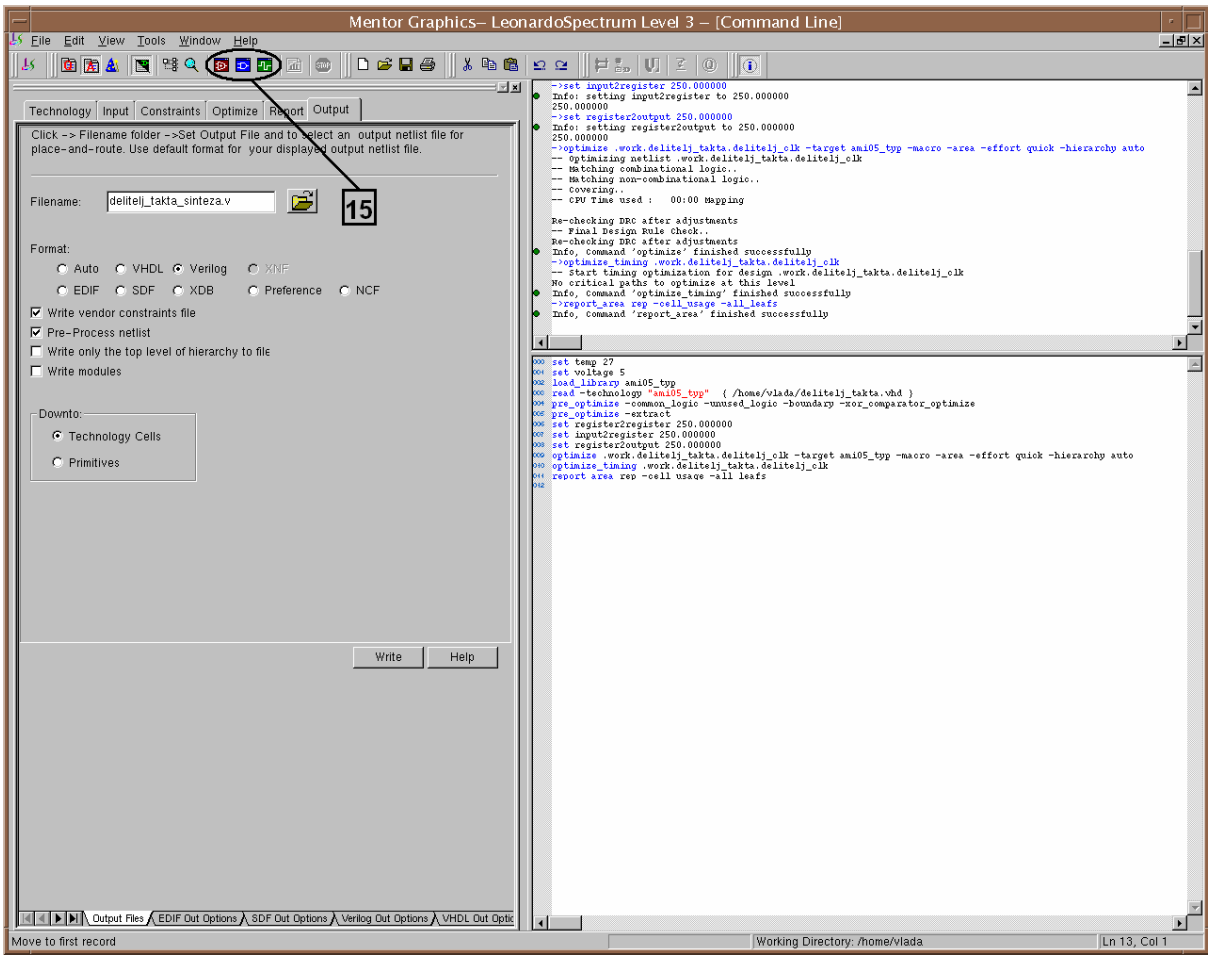


Slika 15. Postavljanje parametara optimizacije

Kreiranje netliste

Na slici 16. predstavljen je poslednji korak sinteze. Potrebno je definisati i imenovati izlazni fajl. Medjutim, potrebno je sačuvati ga kao *Verilog* i kao *VHDL* fajl. *Verilog* fajl će se kasnije koristiti za razmeštaj i povezivanje standardnih ćelija iz biblioteke korišćene tehnologije, dok će se *VHDL* fajl koristiti za simuliranje projekta nakon sinteze tzv. *Post-syntezis simulation*. Dakle, jednostavnim klikom na *write* ovi fajlovi biće upisani u direktorijum koji je naveden na slici 13. kao radni direktorijum. Poželjno je da on bude za jedan stepen iznad *work* direktorijuma, kako se ne bi mešali kodovi makro ćelija, generisane *layout* ćelije i slično.

U zavisnosti od nivoa hijerarhije kola koje se sintetizuje, moguće je definisati kakva će sadržina izlaznog fajla biti, t.j. prilikom generisanja *layout* –a da li će se predstavljati blokovima makroćelija ili tehnološkim ćelijama. U studentskoj verziji (SE) moguće je sagledavanje *layout* – a samo na nivou tehnoloških ćelija.



Slika 16. Postavljanje parametara za izlazne fajlove sinteze

Polje na koje ukazije broj 15, sa slike 16, omogućuje projektantu da sagleda šeme date na nivou tehnoloških ćelija, makroćelija, kao i deo kola gde je najveće kašnjenje signala (u okviru polja 15, gledano sa leva na desno).

Prilikom sintetize složenih projekta, opisanih u više hijerarhijskih nivoa, potrebno je izvršiti sintezu svih hijerarhijski nižih modula, a zatim sintetizovati *top-level* opis sa učitanim svim ostalim makro modulima.

Šeme sintetizovanih kola, koja su korišćena u projektu, «Kalendar» definisan u 6. poglavlju, date su u prilogu B.

Nakon sinteze neophodno je verifikovati rad kola simulacijom. Simulacija nakon sinteze obavlja se na sličan način koji je opisan u odeljku **Simulacija**, s tim što se poziva novo kreirani VHDL ili Verilog fajl.

Nakon uspešne verifikacije rada kola nakon sinteze, potrebno je pripremiti projekat za izradu *layout*–a. Priprema se obavlja pomoću *Design Architect* alata za modifikaciju i generisanje šema. Pored digitalnih kola, u bibliotekama je moguće naći i analogne komponente. Ovaj program podržava i rad sa analognim kolima. Naredno poglavlje posvećeno je opisu primene *Design Architect* alata.

4. Design Architect

Od mnogih mogućnosti koje ovaj program nudi, kao što su simulacija analognih i digitalnih kola, crtanje šema, priprema projekta za dalje korake projektovanja, biće izdvojena ona koja je nama od interesa, a to je priprema za izradu *layout*-a.

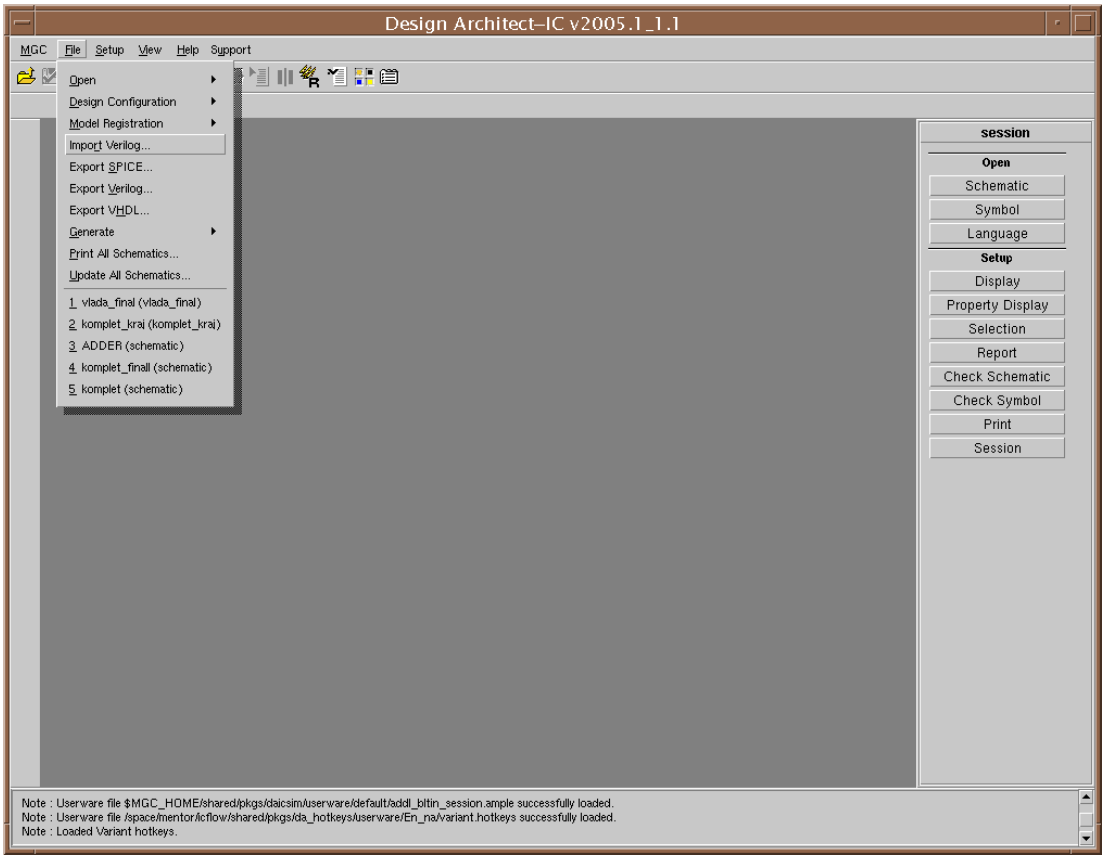
Pozivanje programa

Program se poziva naredbom (prema [Adk1]):

% da_ic

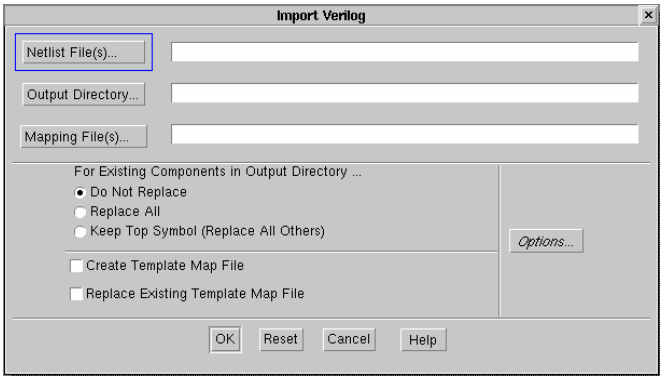
Importovanje Verilog netliste

Način importovanja *Verilog* fajla, dobijenog sintezom prikazan je na slici 17.



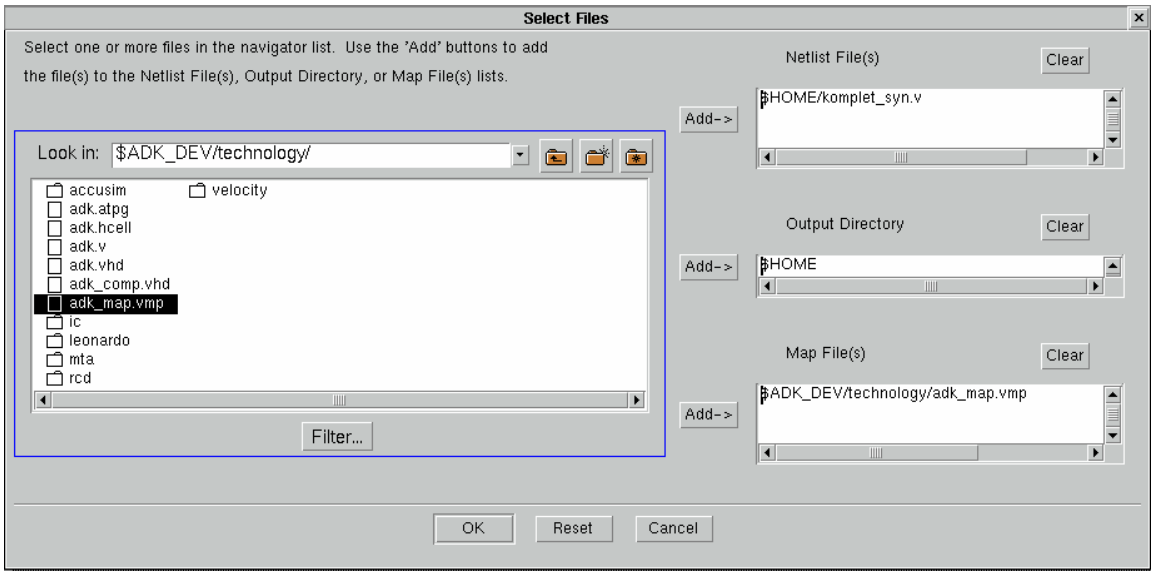
Slika 17. Importovanje *Verilog* fajla u *Design Architect*

Nakon ove akcije pojaviće se *dialog box* dat na slici 18.

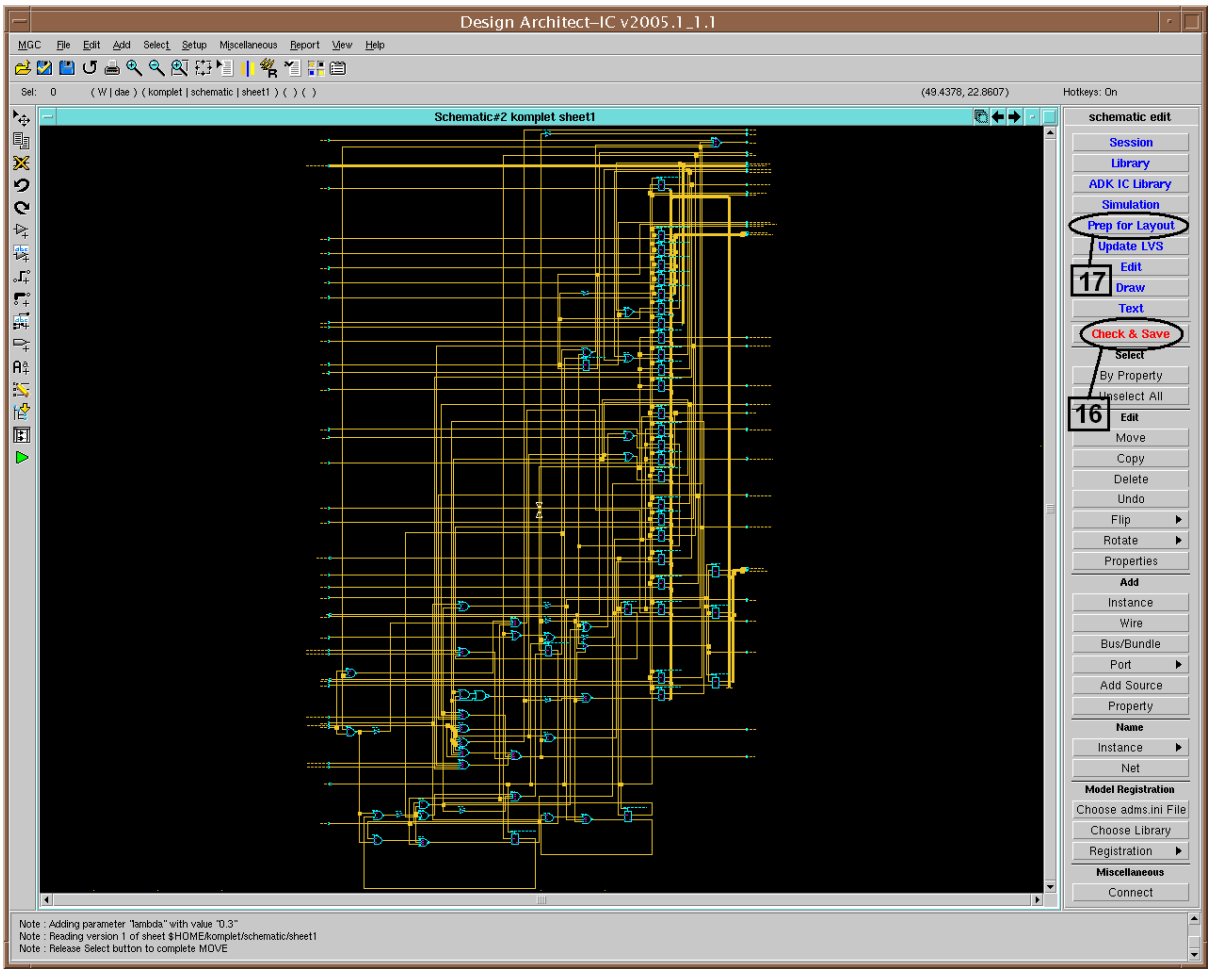


Slika 18. *Dialog box* za učitavanje *verilog* fajla

Nakon klika na plavu oblast sa slike 18. pojaviće se nov *dialog box* dat na slici 19. Dakle, mesto *Verilog* fajla i izlaznog direktorijuma, u kome će se smeštati podaci za generisanje *layout*-a su isti, što se vidi sa slike. Najvažnije je ispravno ukazati na *map* fajl, koji sadrži podatke o načinu povezivanja ćelija, ćije je mesto na hard disku u okviru direktorijuma gde je instaliran *Mentor Graphics* (*space/mentor/adk3_0/ic/techology/).*



Slika 19. Postavljanje parametara za učitavanje *Verilog* fajla

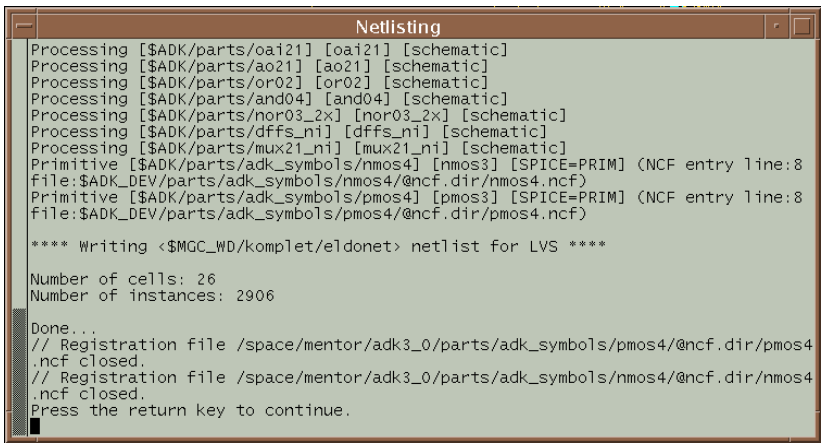


Slika 20. Šema projekta u *Design Architect*-u

Nakon uspešnog učitavanja, šema projekta neće još uvek biti prezentovana. Potrebno je šemu otvoriti selektovanjem *File -> Open -> Schematic*.

Ako je sve prilikom učitavanja *Verilog* fajla bilo dobro podešeno, pojaviće se šema kao na slici 20.

Najpre je potrebno proveriti da li šema zadovoljava sve električne i logičke norme. Provera se startuje klikom na polje označeno brojem 16. Ako je testiranje uspešno izvršeno, pritiskom na polje broj 17. počinje generisanje fajlova za *layout*. Podaci za *layout* editor smeštaju se u navedene pozicije u odgovarajućem *dialog box*-u. Posle uspešnog generisanja net liste za razmeštaj i povezivanje pojaviće se prozor prikazan na slici 21.



Slika 21. Izveštaj o uspešno generisanom *netlist* fajlu

Ovim bi bila generisana *netlista* za *place and route* program, koji se koristi za generisanje *layout*-a čipa. Opširnije čitalac može naći u [Adk1]. U sledećem poglavlju opisani su koraci potrebni za generisanje *layout*-a *IC Station* alatom.

5. IC Station

IC Station predstavlja poslednji alat vezan za projektovanje integrisanih kola u okviru *ASIC Design Kit (ADK)*. Njegova namena u celoj ovoj priči jeste da na osnovu *netlist*-e generiše *layout* čipa koji smo projektovali. Generisanje *layout*-a, obavlja se kroz više koraka, koji će u ovom poglavlju biti objašnjeni. Radi boljeg shvatanja samog procesa i pravila prilikom projektovanja *layout*-a, čitalac se upućuje na [Lit00].

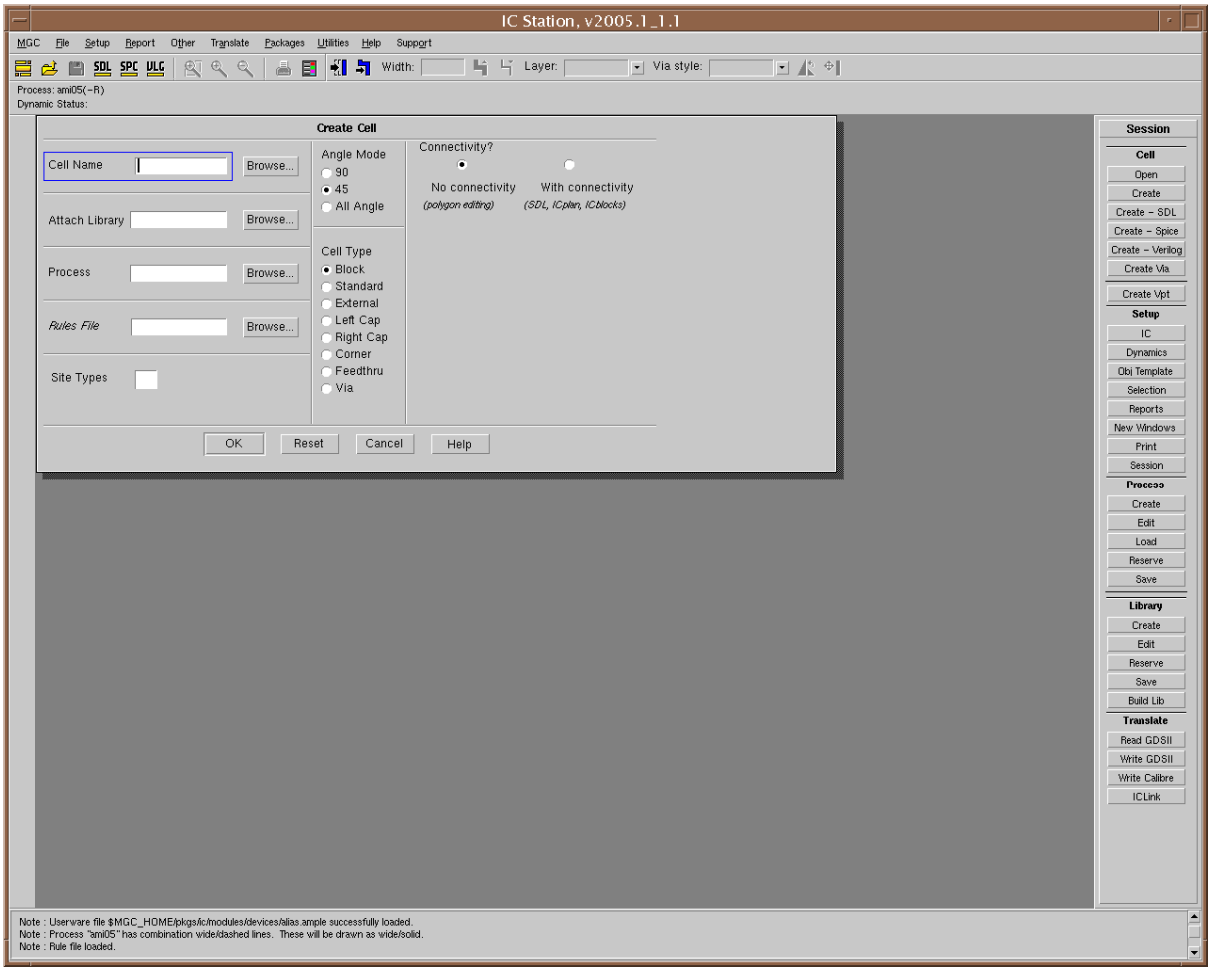
Pozivanje programa

Program se poziva naredbom:

% ic

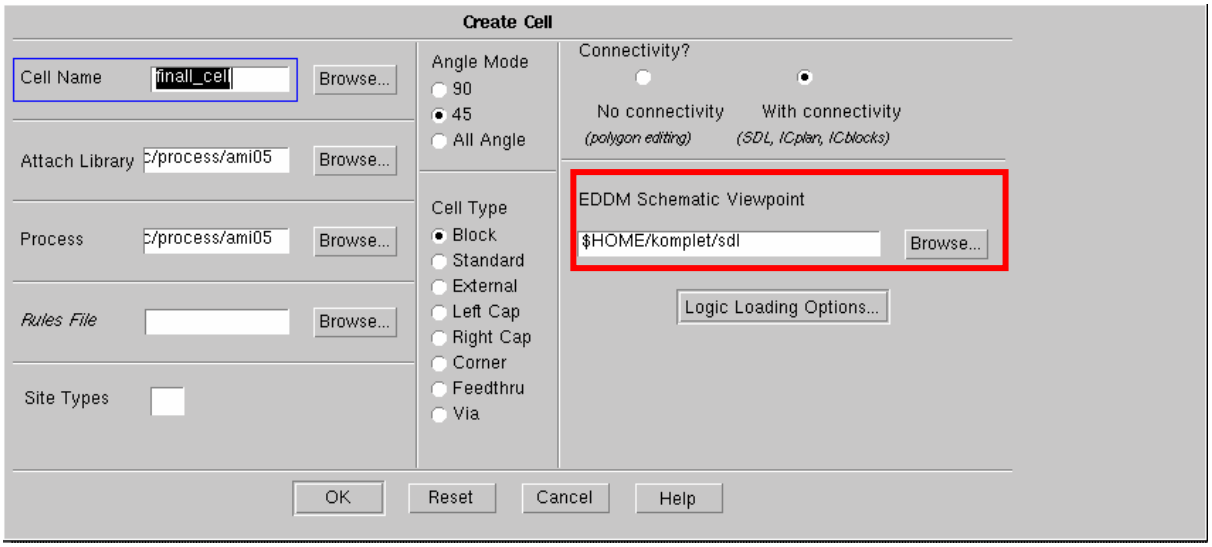
Kreiranje ćelije

Radni prozor programa sa pozvanim *dialog box*-om za kreiranje ćelije dat je na slici 22.



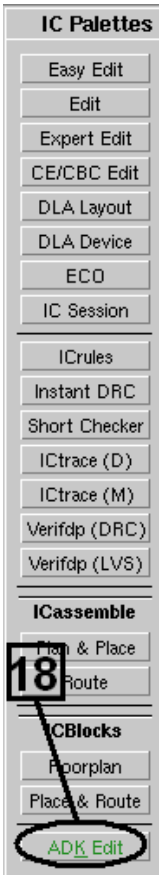
Slika 22. Krerianje makro ćelije

Dialog box je potrebno popuniti na način prikazan na slici 23. Procesi i biblioteke smešteni su u direktorijumu *space/mentor/adk3_0/technology/ic/process/*. U delu vezanom za *viewpoint* (označen sa crvenim poljem na slici 23) jako je važno navesti *sdl* (*Schematic driven Layout*) direktorijum na koji će se *IC* pozivati prilikom generisanja *layout*-a.



Slika 23. Kreiranje makro ćelije sa definisanim parametrima

Nakon kreiranja ćelije pritiskom na OK, otvoriće se blanko prozor. Sada je potrebno aktivirati *adk_edit* paletu iz osnovne palete alata, koja je data na slici 24 (polje obeleženo brojem 18).

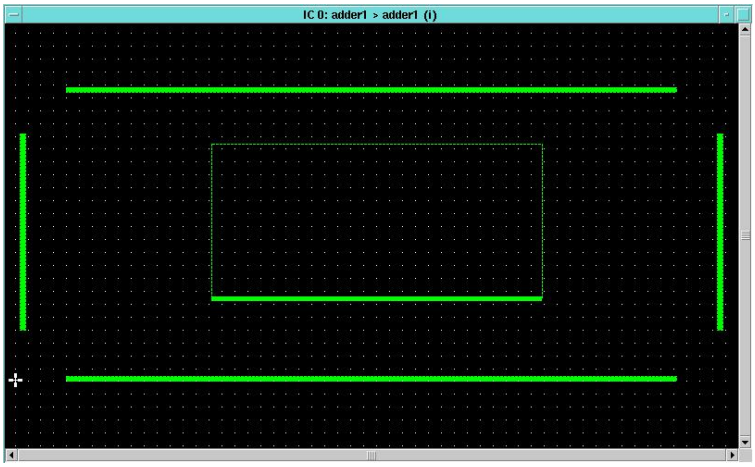


Slika 24. IC paleta alata

Slika 25. Place and route paleta

Planiranje površine čipa

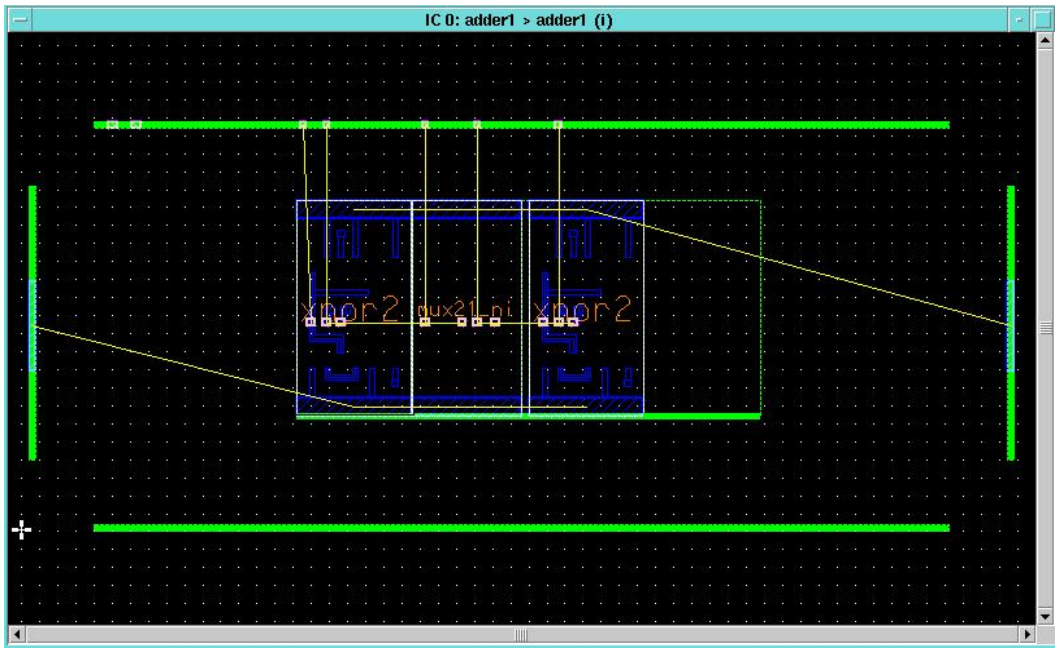
Nakon aktiviranja *adk_edit* palete, koraci koji slede su apsolutno automatizovani. Najpre je potrebno napraviti okvirni plan gde će se smeštati standardne ili makro-ćelije. Selektovanjem *Autofp* sa *place and route* palete, t.j. polja označenog brojem 19 na slici 25, počinje automatsko generisanje plana površine. Nakon toga se na ekranu prikazuje polje sa slike 26.



Slika 26. Generisan *floorplan* makro-ćelije

Postavljanje standardnih ćelija

Sada je potrebno postaviti standardne ćelije. Ovaj korak se izvršava klikom na *StdCel*, polje na koje ukazuje broj 20 sa slike 25. Pored toga potrebno je definisati i portove klikom na *Ports*, polje na koje ukazuje broj 21, sa slike 25. Slika 27. prikazuje popunjen prozor standardnim ćelijama sa portovima.



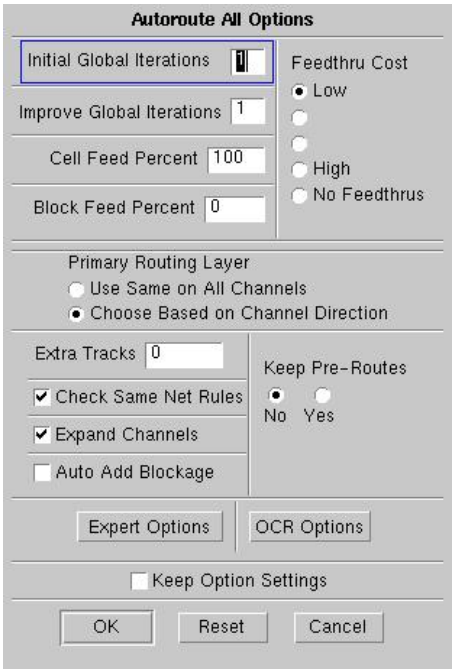
Slika 27. Izgled *layout*-a sa razmeštenim ćelijama i portovima (pre povezivanja)

Povezivanje ćelija

Sledeći korak jeste fizičko povezivanje komponenti kroz više slojeva metalizacije. U primerima koji su obradjeni u ovom radu korišćena su tri sloja metalizacije sa kojih je moguće povezivanje sa spoljnim svetom, dok je za unutrašnja povezivanja u okviru ćelije korišćeno 5 slojeva metala. Potrebno je kliknuti na polje *All*, na koje ukazuje broj 22 sa slike 25. Nakon toga, pojaviće se sledeći *Routing dialog box*.

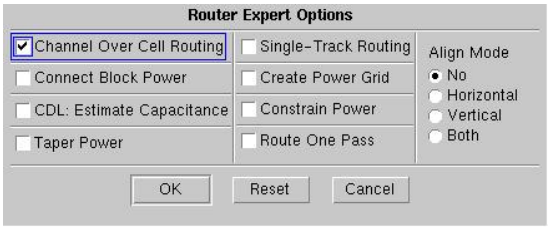
Slika 28. *Routing dialog box*

Klikom na *options* aktivira se novi prozor, u kome je potrebno postaviti odgovarajuće parametre:



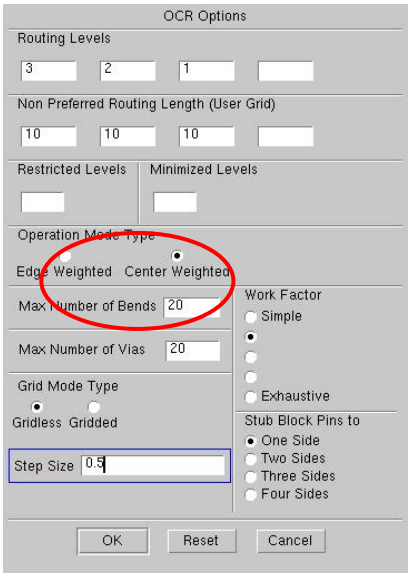
Slika 29. *Options dialog box* za povezivanje komponenti

Klikom na *expert options*, otvara se novi prozor, u kome treba čekirati *channel over cell routing*, kako je i prikazano narednoj slici 30.



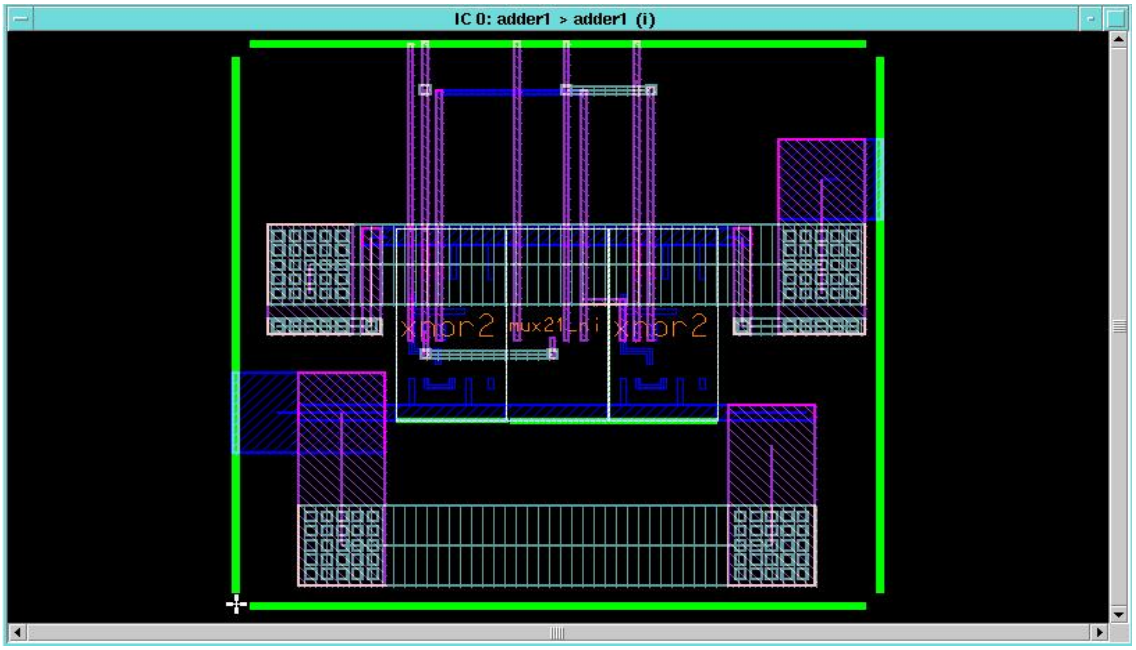
Slika 30. *Dialog box* za postavljanje *expert* parametara *router*-a

Ukoliko se izabere opcija *OCR Options* sa slike 29, pojaviće prozor, prikazan na slici 31 u kome je moguće zadavanje vrednosti parametara.



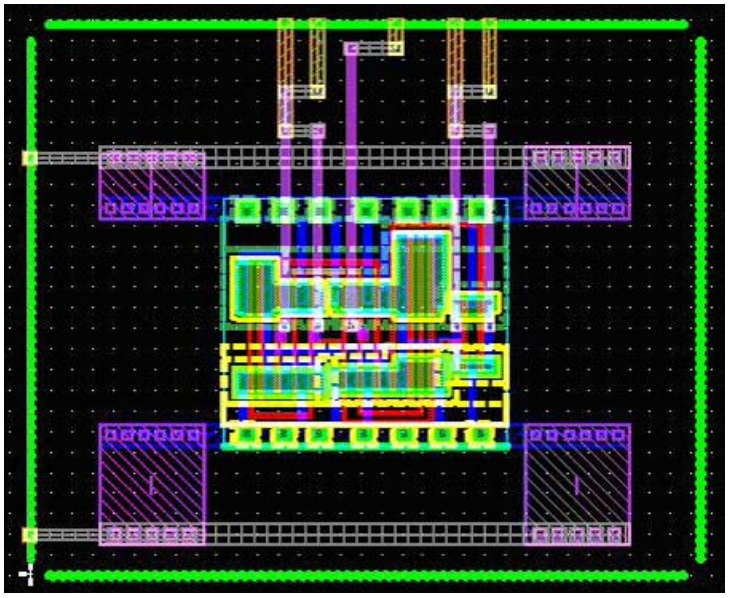
Slika 31. *Dialog box* za setovanje *OCR Options* parametara

Važno je da se *Step Size* postavi na 0.5, što je povezano sa Lambda parametrom [Lit00], kao i *Operation Mode Type* na *Center Weighted.* Potvrdom i zatvaranjem svih gore navedenih *dialog box*-eva, postavljeni su ispravni parametri vezani za povezivanje ćelija korišćenih u projektu. U zavisnosti od veličine projekta, proces povezivanja može potrajati i nekoliko minuta. Slika 32. prikazuje izgled *layouta* ćelije. Najzad, potrebno je izvršiti još jedan korak, a to je testiranje dobijenog *layout*-a. Svaka kockica obojena bledoplavom ili ljubičastom bojom predstavlja *via*-u, t.j. vezu između metala iz različitih slojeva.



Slika 32. *Layout* projektovane ćelije

Moguće je sagledavanje svih pet nivoa metalizacije, koje se aktivira sa *Main* menija odabirom: *context->hierarchy->Peek Area*. Gde je u *dialog box*-u potrebno navesti broj metalizacija koji se želi videti (slika 33.). Prikaz svih generisanih ćejija korišćenih u projektu «Kalendar» nalazi se u prilogu C.



Slika 33. *Layout* projektovane ćelije sa slike 32 sa prikazanim svim nivoima za koje je potrebna izrada maski

Izgled *layout*-a kompletnog kola sa generisanim stopicama dat je u prilogu D. Detaljnije o samim MOS tranzistorima, kao i načinu izrade, dato je u [Lit96]. Na slici 33. može se uočiti oblast obojena zelenom bojom, koja predstavlja N-well, žuta predstavlja P-well, crvena Poly-Si gejt, plava prvi sloj metalizacije, ljubičasta drugi sloj metalizacije itd.

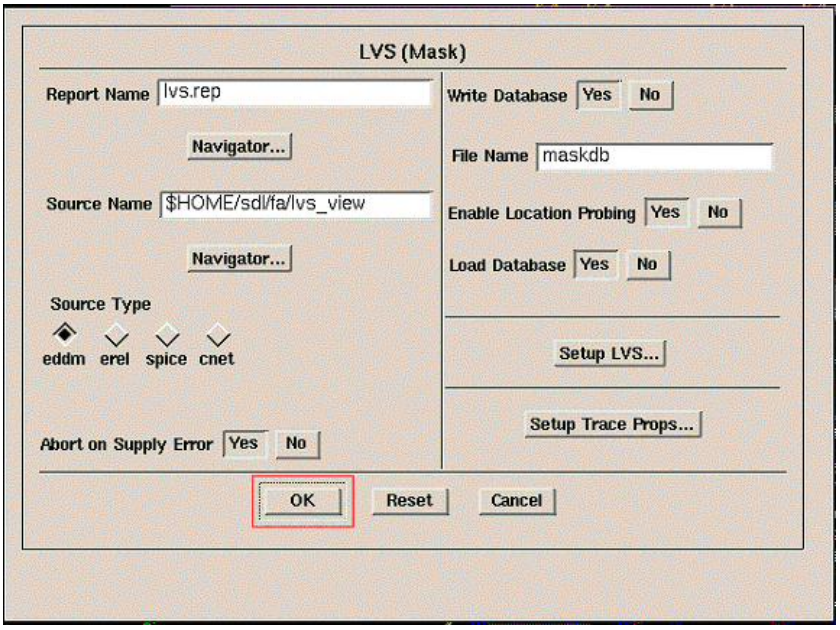
Verifikacija layout-a

Provera pravila projektovanja *Design Rule Check* (DRC), [Lit00, Pet05] predstavlja softverski alat, koji služi da utvrdi činjenicu da li je došlo do prekršenja pravila projektovanja ili ne. Softverski alat za generisanje *layout*-a, koji je u ovom radu opisan, koristi Lambda pravilo projektovanja, gde se dimenzije svake figure na *layout*-u definišu kao celobrojni umnošci parametra Lambda. Međutim, pre DRC testa neophodno je proveriti *overflow* i konzistentnost lejauta, upoređivanjem električne/logičke šeme ekstrahovane na osnovu lejauta sa šematskim, odnosno strukturnim opisom projekta. Ova aktivnost naziva se LVS, *Layout versus Schematic*. Inače *Overflow* predstavlja prekoračenje vezano za sam algoritam povezivanja. U cilju ubrzavanja rada. Tako se najpre povežu ćelije sa najkomlikovanijim vezama (povezivanje se vrši kroz više nivoa). Nakon završetka gore pomenutog povezivanja, neke ćelije ostaju nepovezane, što predstavlja *overflow*.

Postupak verifikacije lejauta počinje izborom opcije *check* sa *Main* menija. Bitno je otkloniti sve uočene prekršaje (*ovefrlow*), koji će se posle testiranja prikazati žutom bojom. Otklanjanje se vrši klikom na polje 23, sa slike 25. i uokvirivanjem odgovarajućeg *overflow*-a.

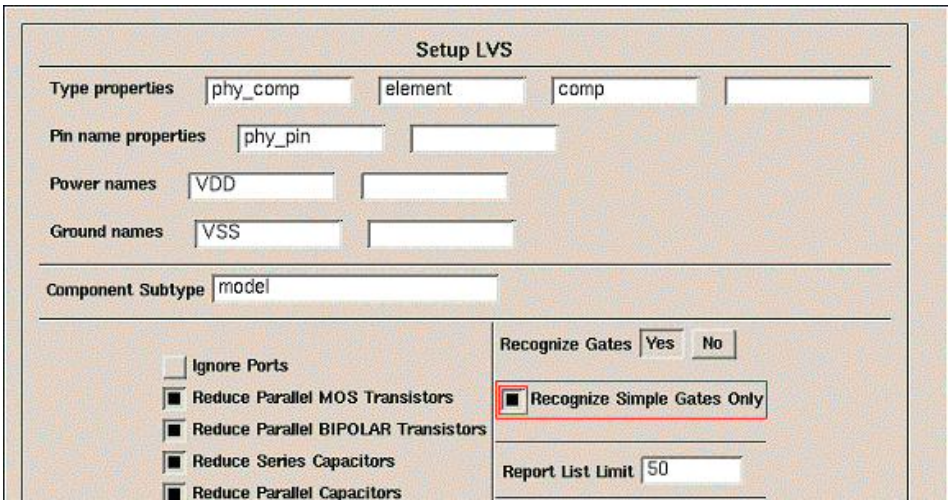
Sledeći korak u okviru verifikacije lejauta odnosi se na LVS i DRC testove [Lit00].

Pokretanje testa se vrši izborom *Verifdp (LVS)* polja u IC paleti sa slike 24. Nakon pokretanja LVS testa pojaviće se *dialog box* prikazan na slici 34. Potrebno je uneti odgovarajuća imena vezana za svaki projekat individualno, t.j. za svaku makroćeliju. Isto tako potrebno je obratiti pažnju na *viewpoint*-te, sa kojima smo se sreli još prilikom priprema sintetizovanog kola za *layout* (potrebno je u okviru *Source name* ukazati na SDL (*Schematic Driven Layout*) *viewpoint*).



Slika 34. *Dialog box* za inicijalizaciju LVS testa

Klikom na *Setup LVS*...u IC paleti sa slike 24, otvara se novi *dialog box* u koga treba uneti parametre date na slikci 35.



Slika 35. Parametri postavljanja LVS testa

Kao rezultat LVS testa dobija se izveštaj o eventualnim greškama vezanim za *layout*, u vidu tekstualnog fajla. Moguće je da se javi više upozorenja jer je u pitanju školska (*Student Edition*) verzija alata za projektovanje integrisanih kola koja ukazuju na greške vezane za *Pad*-ove. Prema [Adk3] ova upozorenja je potrebno ignorisati. Detaljnije o programu, čitalac može naći u [Adk3] i [Adk4].

6. Primer projektovanja ASIC kola metodom standardnih ćelija

Primena *Mentor Graphics* alata za projektovanje integrisanih kola, biće ilustrovana na primeru projektovanja digitalnog kalendara.

6. 1. Projektni zadatak

Projektovati integrisano kolo specifične namene koje obavlja funkciju kalendara sa sledećim osobinama:

- Kolo se pobuđuje osnovnim taktnim signalom frekvencije 32768Hz.
- Kolo na osnovu taktnog signala generiše informaciju o datumu, koja se sastoji od sledećih signala u binarnom obliku [broj bitova dat je u zagradama u formatu (MSB:LSB)]:
  - Dan: (4:0)
  - Mesec: (3:0)
  - Godina: (6:0).
- Kontrolisati različitu dužinu trajanja pojedinih meseci i godina (prestupne).
- Omogućiti reset, preko posebnog ulaza
- Omogućiti preset, odnosno zadavanje početnog datuma kalendara. U ovom slučaju potrebno je ugraditi odgovarajuću kontrolu preko dodatnih pinova.
- Omogućiti efikasno testiranje čipa korišćenjem postojećih pinova.

Posle isteka vremenskog intervala od 24 časa, brojač dana kalendara inkrementira vrednost za jedan. Kalendar mora da uzima u obzir različiti broj dana svakog meseca, da automatski raspoznaje prestupne godine (februar ima 29 dana) počev od nulte kao prestupne.

6.2 Realizacija

Prilikom projektovanja jednog ovakvog čipa potrebno je doneti odluke o osnovnim karakteristikama koje ASIC treba da ima. Na taj način definiše se dobra osnova kako za početak projektovanja arhitekture, tako i za efikasnu podelu poslova po vremenu i resursima (radna snaga i alati koje tim poseduje). S obzirom da je sam zadatak dat u slobodnoj formi, zahtevi mogu da se specificiraju sa velikim stepenom slobode. Iako time projektant dobija veće mogućnosti da se kreativno izrazi, odgovornost koju preuzima je veća, tako da predstavlja veoma ozbiljnu aktivnost.

U ovom slučaju reč je o projektu veoma široke potrošnje, kod koga se ne očekuju problemi vezani za brzinu rada. Kao osnovni kriterijum biće razmatrana cena. Pri tome potrebno je obezbediti pouzdanost i umerenu potrošnju..

Logičko projektovanje

Za veliki broj projektanata najefikasniji način projektovanja predstavlja razbijanje celokupne logike na delove koji se mogu implementirati u vidu konačnih automata. Ovom metodom garantuje se minimalni broj logičkih funkcija za svaki logički blok, čime se direktno utiče na smanjenje složenosti i potrošnje. Međutim, ovakav način projektovanja je podložan greškama.

Drugi pristup projektovanju digitalnih sistema jeste funkcionalna dekompozicija. Funkcionalna dekompozicija omogućuje modularnost digitalnog sistema, što ga čini lakše razumljivim. Nažalost, ovaj pristup projektovanju ne daje optimalnu površinu i kod komplikovanijih sistema moguće su greške usled previda u njihovom međusobnom povezivanju.

Izbor arhitekture

Razmatranjem zadatih specifikacija integrisanog kola kalendara, imajući u vidu da sistem nije mnogo komplikovan, može se doći do zaključka da je jednostavnije kalendar razdvojiti na funkcionalne blokove, odnosno uraditi funkcionalnu dekompoziciju.

Osnovu kola čine tri brojača, slika 36., **brojač_dana, brojač_meseci i brojač_godina**. Svaki od njih koristi digitalne reči različite dužine, saglasno maksimalnom broju, odnosno osnovi brojanja. Brojač dana broji od 1 do 31, meseci od 1 do12, godina od 00 do 99. Ovakvom podelom podržana je preporuka da svaki blok ima jedan taktni signal. Najveću taktnu frekvenciju ima brojač dana. Brojač meseci taktuje se izlazom iz brojača dana, a brojač godina taktuje se izlazom iz brojača meseci.

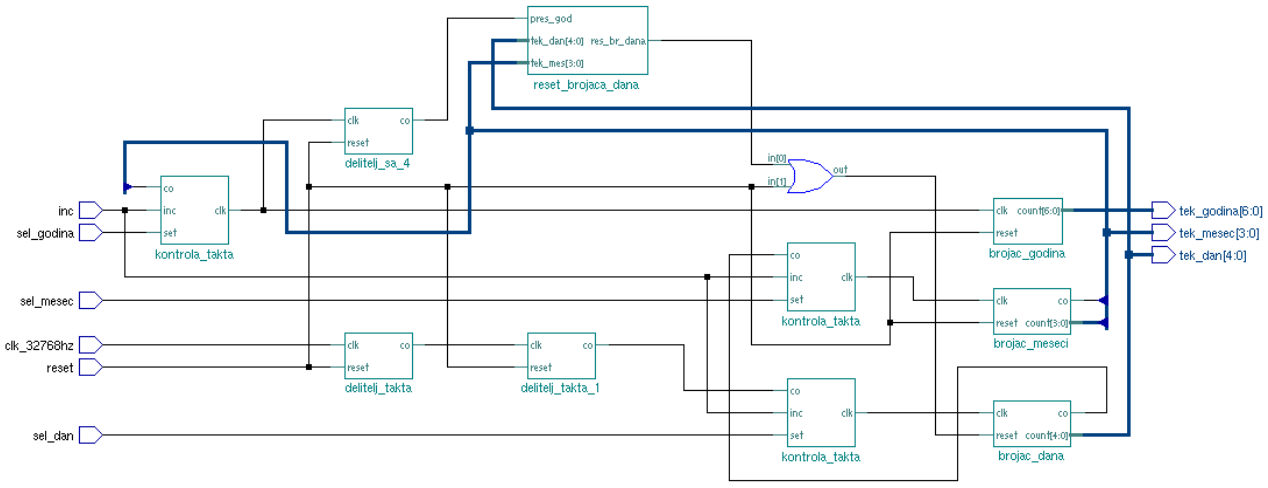
Frekvencija ulaznog taktnog signala mnogo je veća od frekvencije potrebne za taktovanje brojača dana. Zato je potrebno da se osnovni signal takta podeli sa 32768 x 86400. Ovaj delitelj označen je na slici 36. kao blokovi delitelj_takta i delitelj_takta_1. Kontrolu upisa datuma obavljaju blokovi kontrola_takta, koji se na slici 35. javljaju na tri mesta, shodno broju brojača koje je potrebno postaviti. Predviđeno je da se prvo izabere veličina čija se vrednost unosi (datum, meseci ili godine), a zatim inkrementira stanje odgovarajućeg brojača dok se ne dostigne željena cifra. Za razliku od brojača meseci igodina, brojač dana nema fiksnu osnovu brojanja jer se broja dana menja iz meseca u mesec. Zato postoji poseban blok koji obrađuje specifične slučajeve (nejednake dužine pojedinih meseci i prestupne godine). Na slici 35. ovaj blok nazvan je reset_brojača_dana. Na osnovu ovakve arhitekture, mogu se iskristalisati detalji vezani za svaki funkcionalni blok, kao i za signale koji povezuju te blokove.

U naredim koracima projektovanja, potrebno je funkcionalnim opisom definisati gore pomenute blokove, a na kraju *top-level* opisom na nivou arhitekture opisati celokupni projekat. U prilogu A. je tim redosledom i priložena lista kôdova.

7. Rezultati verifikacije

U ovom radu biće prikazani rezultati verifikacije različitih nivoa opisa projekta koji pokazuju ponašanje kalendara pri prelazu kroz karakteristične tačke vezane za prestupne godine kao i način prelaska u novi dan itd.

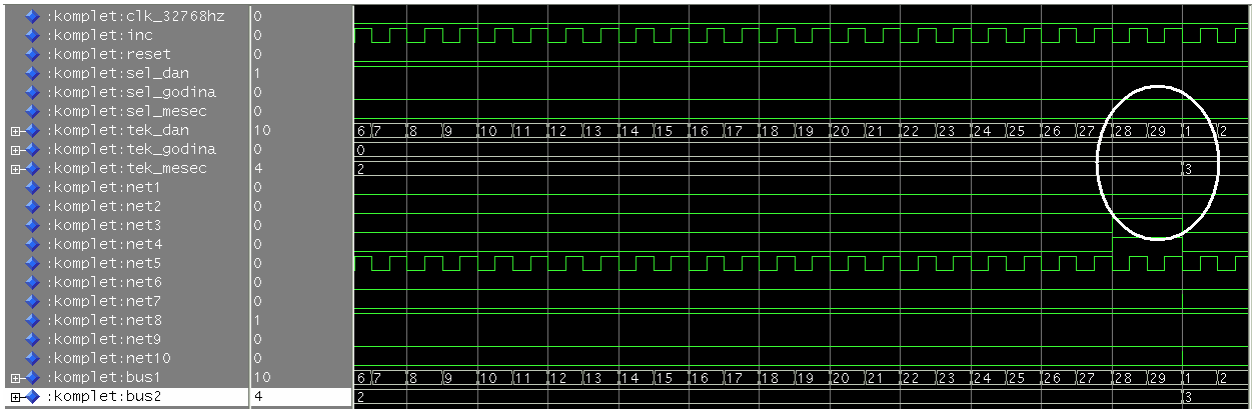
Korisno je napomenuti da je kolo projektovano da bude testabilno [Lit00]. Ovo je učinjeno i iz razloga što su promene jako spore, tako da bi funkcionaklno testiranje u realnom vremenu trajalo izuzetno dugo. Istovremeno, vodilo se računa da ispunjenje zahteva za testabilnošću se ne povećava broj pinova neophodnih za obavljanje osnovne funkcije. Tako koristeći ulazni port ***Inc***, u smislu takt signala može se testirati jedan od selektovanih brojača, koji se selektuju dovodjenjem visokih naponskih nivoa na ulaze ***sel_dan***, ***sel_mesec*** i ***sel_godina***. Pored testiranja, ovi ulazi se koriste i za postavljanje kalendara na željenu vrednost. Radi lakšeg praćenja teksta koji sledi, na slici 36 predstavljena je logička šema kompletnog kalendara.



Slika 36. Šema projektovanog kalendara

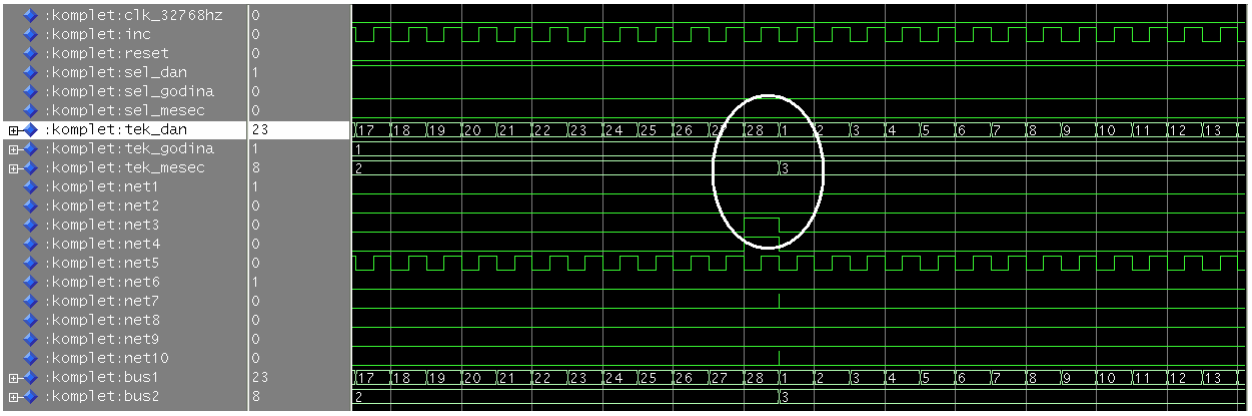
Cilj ovog rada bio je projektovanje čipa, navedenog kalendara. U narednom delu teksta biće predstavljeni dijagrami dobijeni simulacijom ponašanja kalendara u karakterističnim tačkama. Prikazani su dijagrami dobijeni na osnovu verifikacije strukturnog opisa nakon sinteze ali se oni ne razlikuju od rezultata dobijenih simulacijom na nivou funkcionalnog i opisa dobijenog ekstrakcijom iz lejauta. Time je potvrđeno da je projekat uspešno realizovan.

Početno stanje kalendara predstavlja nulta godina koja je usvojena kao prestupna, tako da ima dvadeset devet dana u drugom mesecu. Ovu sizuaciju prikazuje slika 37.



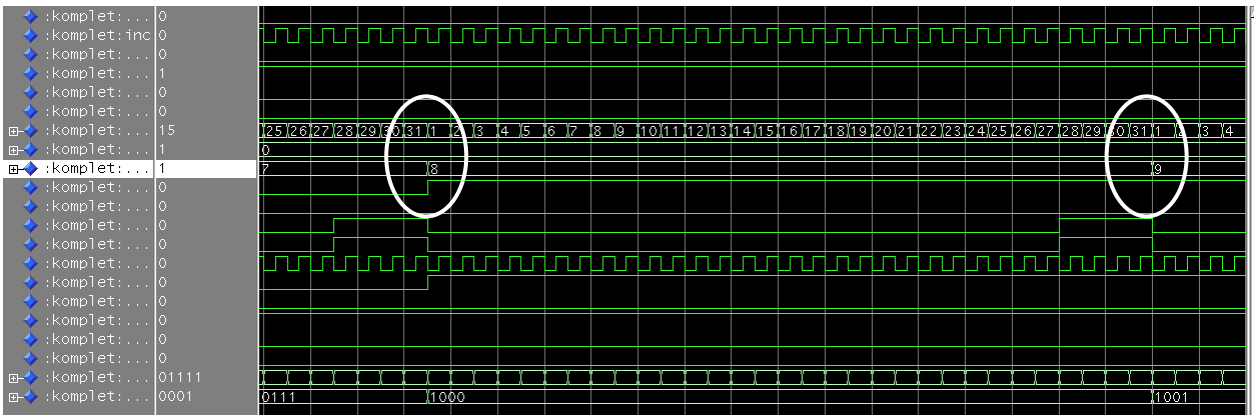
Slika 37. Prikaz rada kalendara u prestupnoj godini

Slika 38. predstavlja prikaz rada kalendara u godini koja nije prestupna. Dakle, ovakva godina je okarakterisana time što drugi mesec ima dvadeset i osam dana.



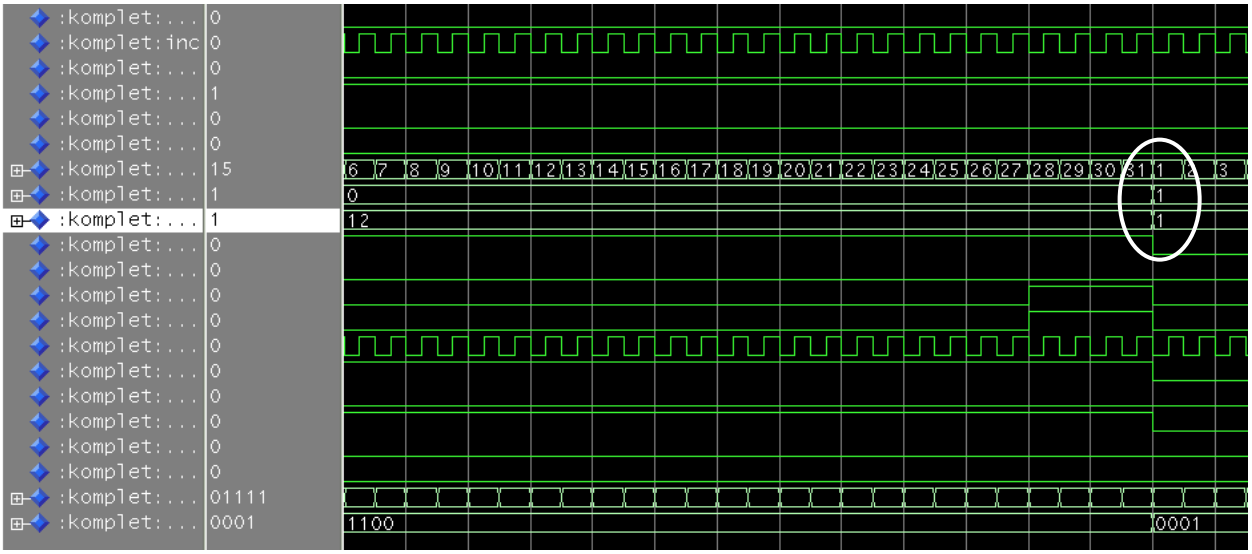
Slika 38. Prikaz rada kalendara u godini koja nije prestupna

Na slici 39. prikazan je prelazak iz jula u avgust, gde se može jasno uočiti da sistem zadržava brojanje brojača dana do 31.



Slika 39. Prikaz prelaska kalendara iz jula u avgust

Na slici 40. prikazan je prelazak u novu godinu.



Slika 40. Prikaz rada kalendara pri prelasku u narednu godinu

Verifikacijom šeme ekstarhovane iz lejauta završen je automatizovani proces projektovanja integrisanih kola.

U prilozima koji slede biće prikazani kodovi VHDL opisa pojedinih modula i kompletnog kola (prilog A), rezultati sinteze (prilog B), genrisani *layout* –svakog modula (prilog C), kao i lejaut kompletnog čipa (prilog D).

8. Zaključak

U radu je opisan postupak projektovanja integrisanog kola kalendara uz pomoć *Mentor Graphics* alata za projektovanje. Kalendar se pobuđuje takt signalom 32768 Hz. Moguće je postavljanje datuma dovođenjem impulsa sa tastera na ulazni port ***Inc*** uz selektovanje dana, meseca ili godine.

U prva pet poglavlja ovog rada data su objašnjenja vezana za korišćenje potrebnih alata za projektovanje ASIC čipa. Objašnjenja koja su data nisu detaljna. Detaljnija objašnjenja, mogu se naći u korišćenoj literaturi.

U šestom i sedmom poglavlju data su objašnjenja vezana za sam projekat, koja je moguće adekvatno pratiti zahvaljujući detaljnim prilozima.

Projekat je baziran na sledećih nekoliko faza:

- Funkcionalni opis i simulacija svih modula pomoću *ModelSim* alata
- Strukturni opis i njegova simulacija pomoću *modelSim* alata
- Sinteza svih modula kao i *top-level* opisa korišćenjem alata *Leonardo Spectrum*
- Importovanje Verilog netiste i priprema *layout*-a svih modula kao i *top-level* opisa korišćenjem *DesignArchitect* alata.
- Generisanje *layout*-a svih modula, kao i *top-level* entiteta korišćenjem *ICStation* alata
- Ubacivanje stopica kompletnom čipu
- Testiranje projekta.

Tokom rada sa pomenutim alatima javljali su se odredjeni problemi. Neki od njih su:

- Prilikom rada sa *ModelSim* alatom bilo je problema oko aktiviranja VHDL opisa u okviru željenog projekta. Problem se konkretno odnosi na neprepoznavanje opisa, čime je bilo nemoguće vršiti kompajliranje i simulaciju. Problem je prevaziđen generisanjem VHDL fajla i naknadnim importovanjem u listu aktivnih kôdova za dati projekat
- Problem sa ubacivanjem stopica rešen je nalaženjem opcije u alatu za sintezu, gde je inače i potrebno definisati da li će se čipu dodavati stopice.
- Veoma ozbiljan problem javio se prilikom potrebe za importovanjem Verilog netliste u *Design Architect*. Problem je rešen pravilnim postavljanjem parametara u *locatin_map.adk* fajlu.
- Zamerka je što nije moguće *top-level layout* sagledati na nivou generisanih makro-ćelija, kao i nemogućnost imenovanja stopica.
- Alati su isuviše automatizovani, što može projektantu zadati dosta problema kada je potrebno neku akciju izvesti ručno. Na primer raspored stopica, njihove veze itd.

Pored nedostataka, potrebno je naglasiti da pomenuti alati za ASIC dizajn imaju relativno jednostavne načine uklanjanja greški, koje su vezane za pravila projektovanja.

Specifikacije projektovanog čipa:

Napajanje: 5V
Tehnologija: AMI 0.5, λ=0.5 μm, širina poly-Si gejta 2λ
Aktivna površina čipa: 1.1005 mm X 1.05175 mm
Frekvencija definisana prilikom sinteze: 4 MHz

9. Literatura

[Adk1] Designing ASICs with the ADK Design Kit and Mentor Graphics Tools, User Manual for ADK Design Kit Version 1.6., *Mentor Graphics 2005.*

[Adk2] Model*Sim* Advanced Verification and Debuging SE Tutorial, Version 6.0.c. January 21, 2005. Mentor Graphics Corporation 8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777

[Adk3] Placing I/O Pads.htm, User Manual for IC Station, *Mentor Graphics*. Mentor Graphics Corporation 8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.

[Adk4] http://www.scudc.scu.edu/mentortu/mg_sc.html#INTRO, Help Files for Place and Route.

[Lit96] Dr. Vančo B. Litovski, Dr Slobodan M. Lazović, "OSNOVI ELEKTRONIKE**",** Čuperak Plavi, Elektronski Fakultet Niš, Prosveta, Niš 1996.

[Lit00] Dr. Vančo B. Litovski, "PROJEKTOVANJE ELEKTRONSKIH KOLA: SIMULACIJA, OPTIMIZACIJA, TESTIRANJE, FIZIČKO PROJEKTOVANJE", I izdanje, DGIP "Nova Jugoslavija" , Vranje, 2000.

[Pet05] Dr. Predrag P. Petković, Mr. Miljana Sokolović i Mr. Bojan Andjelković, "PROJEKTOVANJE INTEGRISANIH KOLA – VHDL simulacija i sinteza", Elektronski fakultet u Nišu u saradnji sa Austrian Cooperation Eastern Europe WUS Austria, interna publikacija, Niš, 2005.

Prilog A. VHDL kôdovi kalendara

Opis bafera dana:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : bafer_dana
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11
12  entity buffer_dana is
13  port(ulaz:in std_logic_vector(4 downto 0);
14      izlaz:out std_logic_vector(4 downto 0));
15  end entity buffer_dana;
16
17  architecture buff_dana of buffer_dana is
18  begin
19  izlaz<=ulaz;
20  end architecture buff_dana;
```

Opis bafera meseci:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : bafer_meseci
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11
12  entity buffer_meseci is
13  port(ulaz:in std_logic_vector(3 downto 0);
14      izlaz:out std_logic_vector(3 downto 0));
15  end entity buffer_meseci;
16
17  architecture buff of buffer_meseci is
18  begin
19  izlaz<=ulaz;
20  end architecture buff;
```

Opis brojača dana:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : brojac_dana
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11  use IEEE.std_logic_unsigned.all;
12
13  entity brojac_dana is
14  port(clk:in std_logic;
15      reset:in std_logic;
16      co:out std_logic;
17      count:out std_logic_vector(4 downto 0));
18  end entity brojac_dana;
19  architecture brojac_dana of brojac_dana is
20
21  begin
22  process (clk, reset) is
23      variable brojilo:std_logic_vector(4 downto 0);
24  begin
25      if reset='1' then
26          brojilo:="00001";
27      elsif clk='0' and clk'event then
28          if brojilo="11111" then
29              brojilo:="00001";
30          else
31              brojilo:=brojilo + 1;
32          end if;
33      end if;
34      co<=brojilo(4) and brojilo(3) and brojilo(2);
35      count<=brojilo;
36  end process;
37  end architecture brojac_dana;
```

Opis brijača meseci:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : Br_mes
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11  use IEEE.std_logic_unsigned.all;
12
13  entity brojac_meseci is
14  port(clk:in std_logic;
15      reset:in std_logic;
16      co:out std_logic;
17      count:out std_logic_vector(3 downto 0));
18  end entity brojac_meseci;
19  architecture brojac_meseci of brojac_meseci is
20
21  begin
22  process (clk, reset) is
23      variable counter:std_logic_vector(3 downto 0);
24  begin
25      if reset='1' then
26          counter:="0001";
27      elsif clk='0' and clk'event then
28          if counter="1100" then
29              counter:="0001";
30          else
31              counter:=counter + 1;
32          end if;
33      end if;
34      co<=counter(3);
35      count<=counter;
36  end process;
37
38
39  end architecture brojac_meseci;
```

Opis brojača godina:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : brojac_godina
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11  use IEEE.std_logic_unsigned.all;
12
13  entity brojac_godina is
14  port(clk:in std_logic;
15      reset:in std_logic;
16      count:out std_logic_vector(6 downto 0));
17  end entity brojac_godina;
18
19  architecture br_godina of brojac_godina is
20  begin
21  process(clk, reset) is
22      variable counter:std_logic_vector(6 downto 0);
23  begin
24      if reset='1' then
25          counter:="0000000";
26      elsif clk'event and clk='0' then
27          if counter="1100011" then
28              counter:="0000000";
29          else
30              counter:=counter + 1;
31          end if;
32      end if;
33      count<=counter;
34  end process;
35  end architecture br_godina;
```

Opis delitelja sa 4:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : delitelj_sa_4
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11  use IEEE.std_logic_unsigned.all;
12
13  entity delitelj_sa_4 is
14  port(clk:in std_logic;
15      reset:in std_logic;
16      co:out std_logic);
17  end entity delitelj_sa_4;
18
19  architecture delitelj4 of delitelj_sa_4 is
20  signal counter:std_logic_vector(1 downto 0);
21  begin
22  process(clk, reset) is
23  begin
24      if reset='1' then
25          counter<="00";
26      elsif clk='1' and clk'event then
27          if counter="11" then
28              counter<="00";
29          else
30              counter<=counter +1;
31          end if;
32      end if;
33      co<=(not counter(1)) and (not counter(0));
34  end process;
35  end architecture delitelj4;
```

Opis delitelja takta:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : delitelj_takta
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11  use IEEE.std_logic_unsigned.all;
12
13  entity delitelj_takta is
14  port(clk:in std_logic;
15      reset:in std_logic;
16      co:out std_logic);
17  end entity delitelj_takta;
18
19  architecture delitelj_clk of delitelj_takta is
20  signal counter:std_logic_vector(15 downto 0);
21  begin
22  process(clk, reset) is
23  begin
24      if reset='1' then
25          counter<="0000000000000000";
26      elsif clk='1' and clk'event then
27          if counter="1000000000000000" then
28              counter<="0000000000000000";
29          else
30              counter<=counter + 1;
31          end if;
32      end if;
33      co<=counter(15);
34  end process;
35
36  end architecture delitelj_clk;
```

Opis delitelja takta 1:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : delitelj_takta_1
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11  use IEEE.std_logic_unsigned.all;
12
13  entity delitelj_takta_1 is
14  port(clk:in std_logic;
15      reset:in std_logic;
16      co:out std_logic);
17  end entity delitelj_takta_1;
18
19  architecture delitelj_clk_1 of delitelj_takta_1 is
20  signal counter:std_logic_vector(16 downto 0);
21  begin
22  process(clk, reset) is
23  begin
24      if reset='1' then
25          counter<="00000000000000000";
26      elsif clk='1' and clk'event then
27          if counter="10101000110000000" then
28              counter<="00000000000000000";
29          else
30              counter<=counter + 1;
31          end if;
32      end if;
33      co<=counter(16);
34  end process;
35  end architecture delitelj_clk_1;
```

Opis kola za reset brojača dana:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : kolo_za_reset_brojaca_dana
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11
12  entity reset_brojaca_dana is
13  port(pres_god:in std_logic;
14      tek_dan:in std_logic_vector(4 downto 0);
15      tek_mes:in std_logic_vector(3 downto 0);
16      res_br_dana:out std_logic);
17  end entity reset_brojaca_dana;
18
19  architecture r_b_d of reset_brojaca_dana is
20  begin
21  p0:process(pres_god, tek_mes, tek_dan) is
22  begin
23      if pres_god='1' and tek_mes="0010" and tek_dan="11110" then
24
25          res_br_dana<='1';
26
27      elsif pres_god='0' and tek_mes="0010" and tek_dan="11101" then
28
29          res_br_dana<='1' ;
30
31      elsif (tek_mes="0100" or tek_mes="0110" or tek_mes="1001" or
32          tek_mes="1011") and tek_dan="11111" then
33
34          res_br_dana<='1' ;
35
36      elsif (tek_mes="0001" or tek_mes="0011" or tek_mes="0101" or
37          tek_mes="0111" or tek_mes="1000" or
38          tek_mes="1010" or tek_mes="1100") and
39          tek_dan="00000" then
40
41          res_br_dana<='1' ;
42
43      else
44          res_br_dana<='0';
45      end if;
46  end process p0;
47  end architecture r_b_d;
```

Opis kola za kontrolu takta:

```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : kolo_za_kontrolu_takta
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11
12  entity kontrola_takta is
13  port(co:in std_logic;
14       inc:in std_logic;
15       set:in std_logic;
16       clk:out std_logic);
17  end entity kontrola_takta;
18
19  architecture kontrola_takta of kontrola_takta is
20  signal net1:std_logic;
21  signal net2:std_logic;
22  begin
23  net1<=inc and set;
24  clk<=net1 or net2;
25  net2<=not(set) and co;
26  end architecture kontrola_takta;
```

U daljem delu ovog poglavlja sledi kôd koji predstavlja opis projekta na najvišem hijerarhijskom nivou. U njemu su definisane veze izmedju prethodno opisanih komponenata.

Opis kompletnog kalendara:

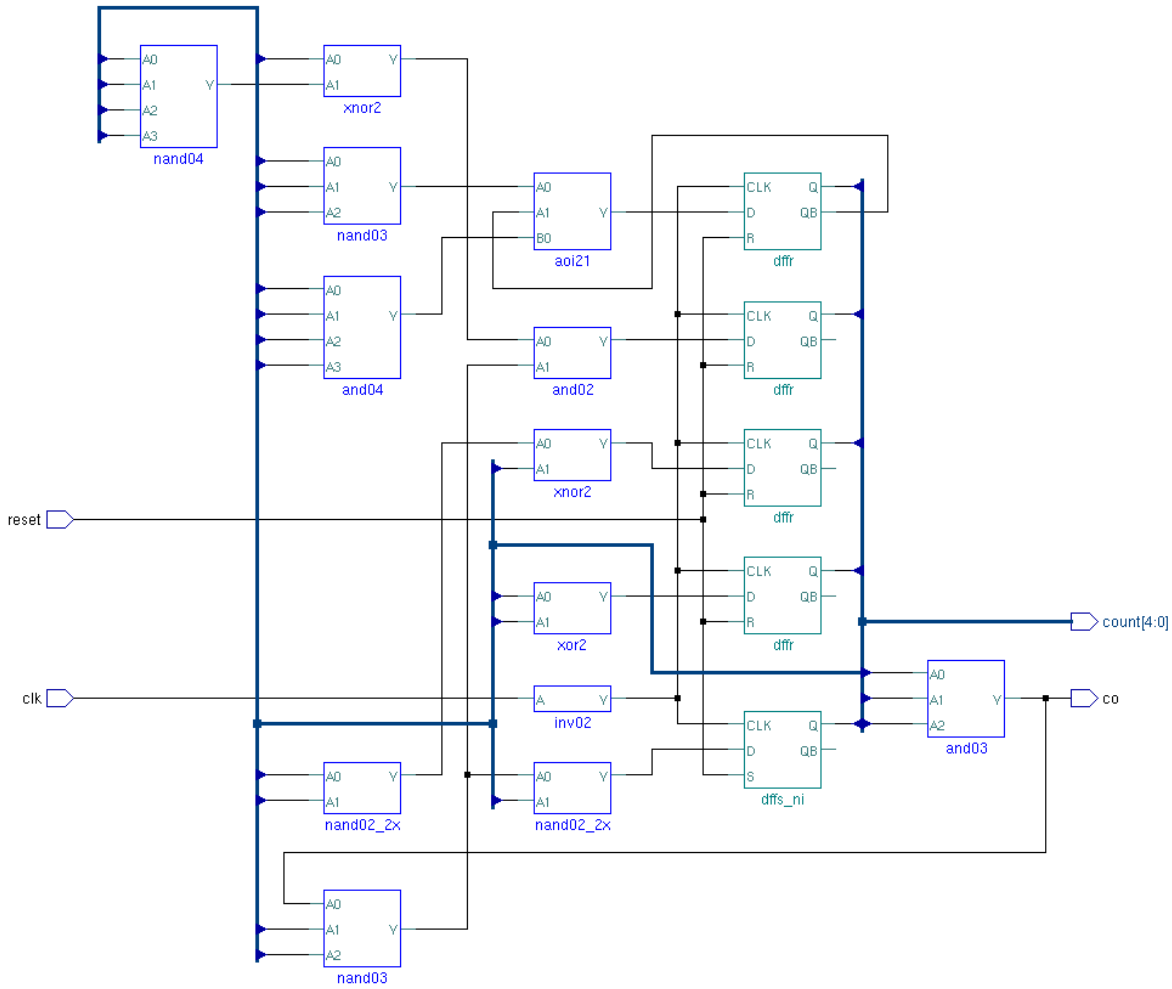
```
1   ----------------------------------------------------------------------
2   --
3   -- Title       : kalendar
4   -- Design      : Diplomski rad
5   -- Author      : Vladimir Petrovic
6   -- Company     : Elektronski fakultet Nis - LEDA
7   --
8   ----------------------------------------------------------------------
9   library IEEE;
10  use IEEE.std_logic_1164.all;
11
12  entity komplet is
13     port(clk_32768hz:in std_logic;
14          inc:in std_logic;
15          reset:in std_logic;
16          sel_dan:in std_logic;
17          sel_godina:in std_logic;
18          sel_mesec:in std_logic;
19          tek_dan:out std_logic_vector(4 downto 0);
20          tek_godina:out std_logic_vector(6 downto 0);
21          tek_mesec:out std_logic_vector(3 downto 0));
22  end entity komplet;
23
24  architecture komplet of komplet is
25     --Deklaracija komponeneti--
26     component brojac_dana
27          port(clk:in std_logic;
28               reset:in std_logic;
29               count:out std_logic_vector(4 downto 0);
30               co:out std_logic);
31     end component;
32     component brojac_godina
33          port(clk:in std_logic;
34               reset:in std_logic;
35               count:out std_logic_vector(6 downto 0));
36     end component;
37     component brojac_meseci
38          port(clk:in std_logic;
39               reset:in std_logic;
40               co:out std_logic;
41               count:out std_logic_vector(3 downto 0));
42     end component;
43     component buffer_dana
44          port(ulaz:in std_logic_vector(4 downto 0);
45               izlaz:out std_logic_vector(4 downto 0));
46     end component;
47     component buffer_meseci
48          port(ulaz:in std_logic_vector(3 downto 0);
49               izlaz:out std_logic_vector(3 downto 0));
50     end component;
51     component delitelj_sa_4
52          port(clk:in std_logic;
53               reset:in std_logic;
54               co:out std_logic);
55     end component;
56     component delitelj_takta
57          port(clk:in std_logic;
```

```
58              reset:in std_logic;
59              co:out std_logic);
60      end component;
61      component delitelj takta 1
62          port(clk:in std_logic;
63              reset:in std_logic;
64              co:out std_logic);
65      end component;
66      component kontrola_takta
67          port(co:in std_logic;
68              inc:in std_logic;
69              set:in std_logic;
70              clk:out std_logic);
71      end component;
72      component reset_brojaca_dana
73          port(pres_god:in std_logic;
74              tek_dan:in std_logic_vector(4 downto 0);
75              tek_mes:in std_logic_vector(3 downto 0);
76              res_br_dana:out std_logic);
77      end component;
78      --Deklaracija signala--
79      signal net1:std_logic;
80      signal net2:std_logic;
81      signal net3:std_logic;
82      signal net4:std_logic;
83      signal net5:std_logic;
84      signal net6:std_logic;
85      signal net7:std_logic;
86      signal net8:std_logic;
87      signal net9:std_logic;
88      signal net10:std_logic;
89      signal bus1:std_logic_vector(4 downto 0);
90      signal bus2:std_logic_vector(3 downto 0);
91  begin
92      u1:brojac_dana
93      port map(clk=>net5,
94          co=>net4,
95          count=>bus1,
96          reset=>net7);
97      u2:brojac_meseci
98      port map(clk=>net3,
99          co=>net1,
100         count=>bus2,
101         reset=>reset);
102     u3:brojac_godina
103     port map(clk=>net6,
104         count=>tek_godina,
105         reset=>reset);
106     u4:kontrola_takta
107     port map(clk=>net3,
108         co=>net4,
109         inc=>inc,
110         set=>sel_mesec);
111     u5:kontrola takta
112     port map(clk=>net6,
113         co=>net1,
114         inc=>inc,
```

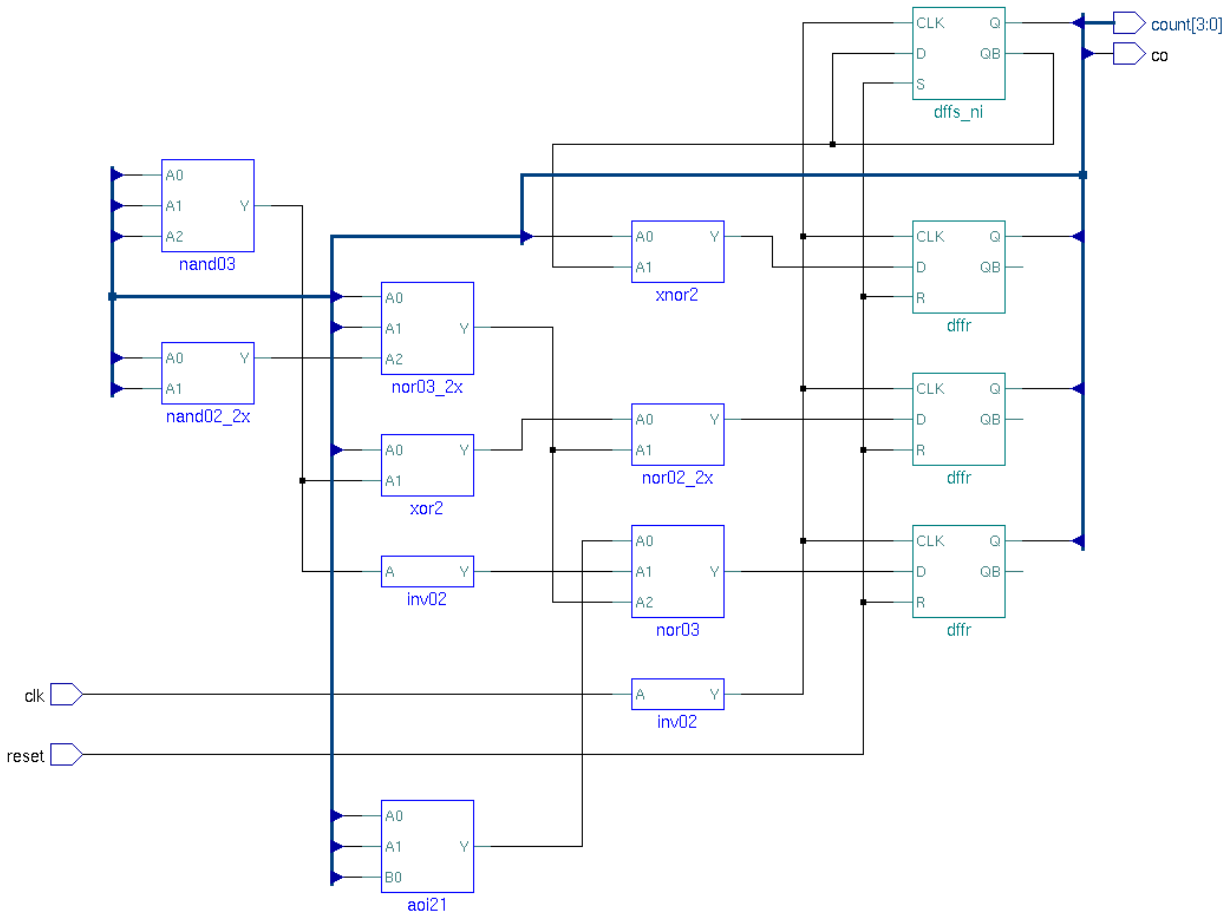
```
115         set=>sel_godina);
116     u6:kontrola_takta
117     port map(clk=>net5,
118         co=>net2,
119         inc=>inc,
120         set=>sel_dan);
121     u7:delitelj_sa_4
122     port map(clk=>net6,
123         co=>net8,
124         reset=>reset);
125     u8:reset_brojaca_dana
126     port map(pres_god=>net8,
127         res_br_dana=>net10,
128         tek_dan=>bus1,
129         tek_mes=>bus2);
130
131     net7<=net10 or reset;
132     u9:delitelj_takta
133     port map(clk=>clk_32768hz,
134         co=>net9,
135         reset=>reset);
136     u10:delitelj_takta_1
137     port map(clk=>net9,
138         co=>net2,
139         reset=>reset);
140     u11:buffer_dana
141     port map(izlaz=>tek_dan,
142         ulaz=>bus1);
143     u12:buffer_meseci
144     port map(izlaz=>tek_mesec,
145         ulaz=>bus2);
146 end architecture komplet;
```

8. Prilog B. Šeme sintetizovanih modula kalendara

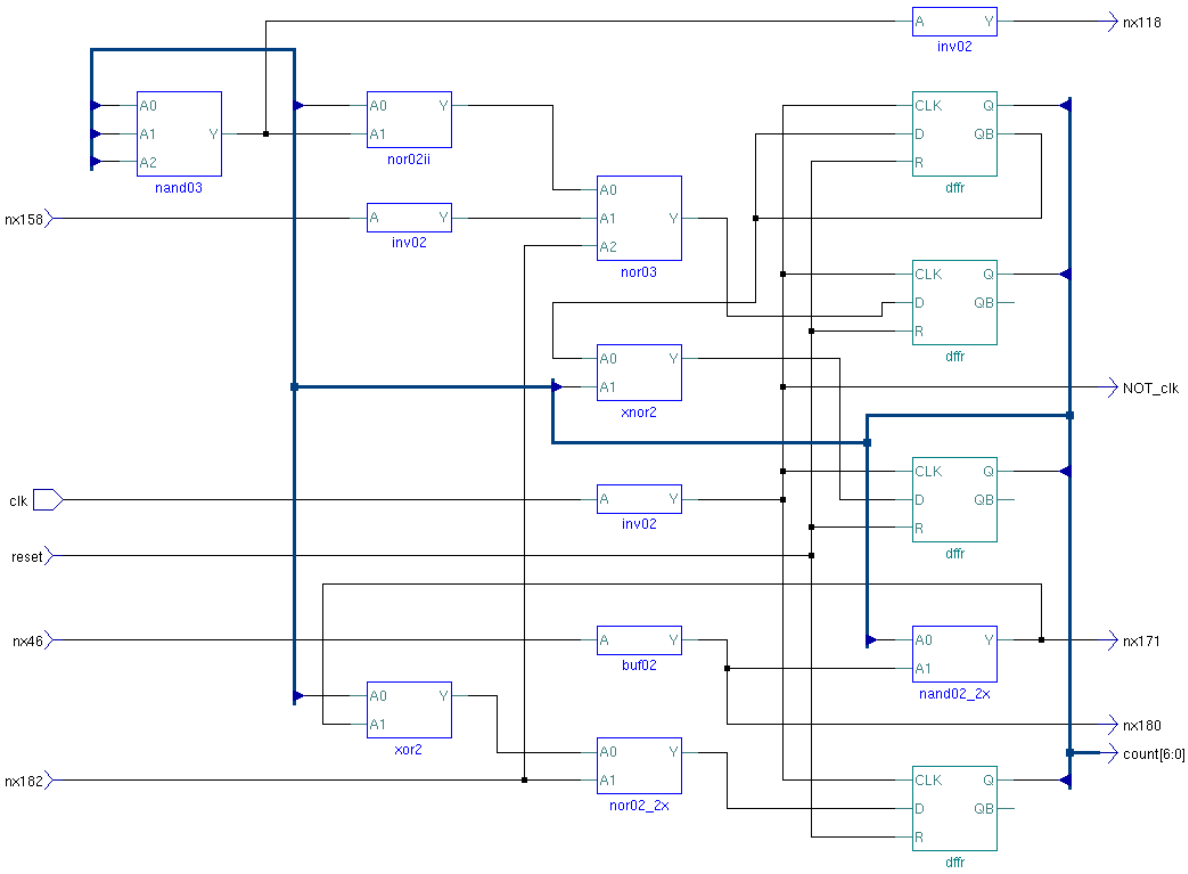
Šema sintetizovanog kola brojača dana:



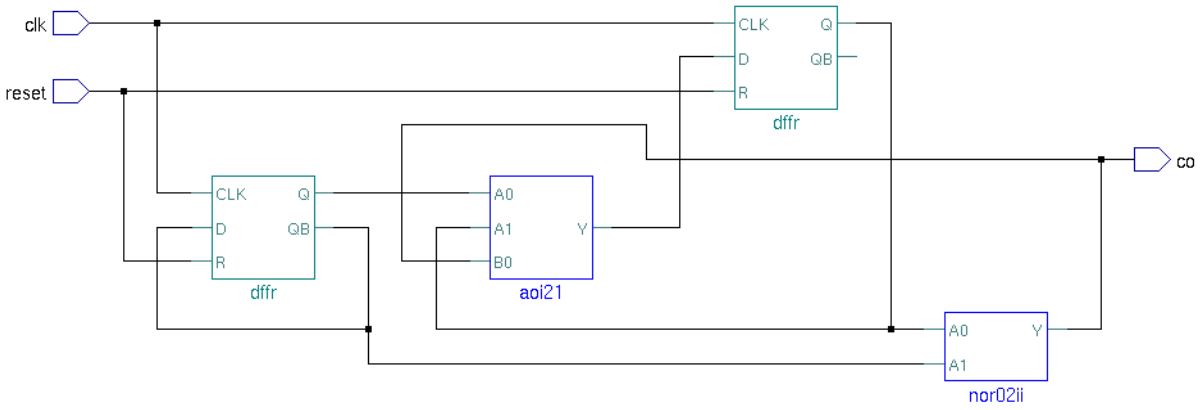
Šema sintetizovanog kola brojača meseci:



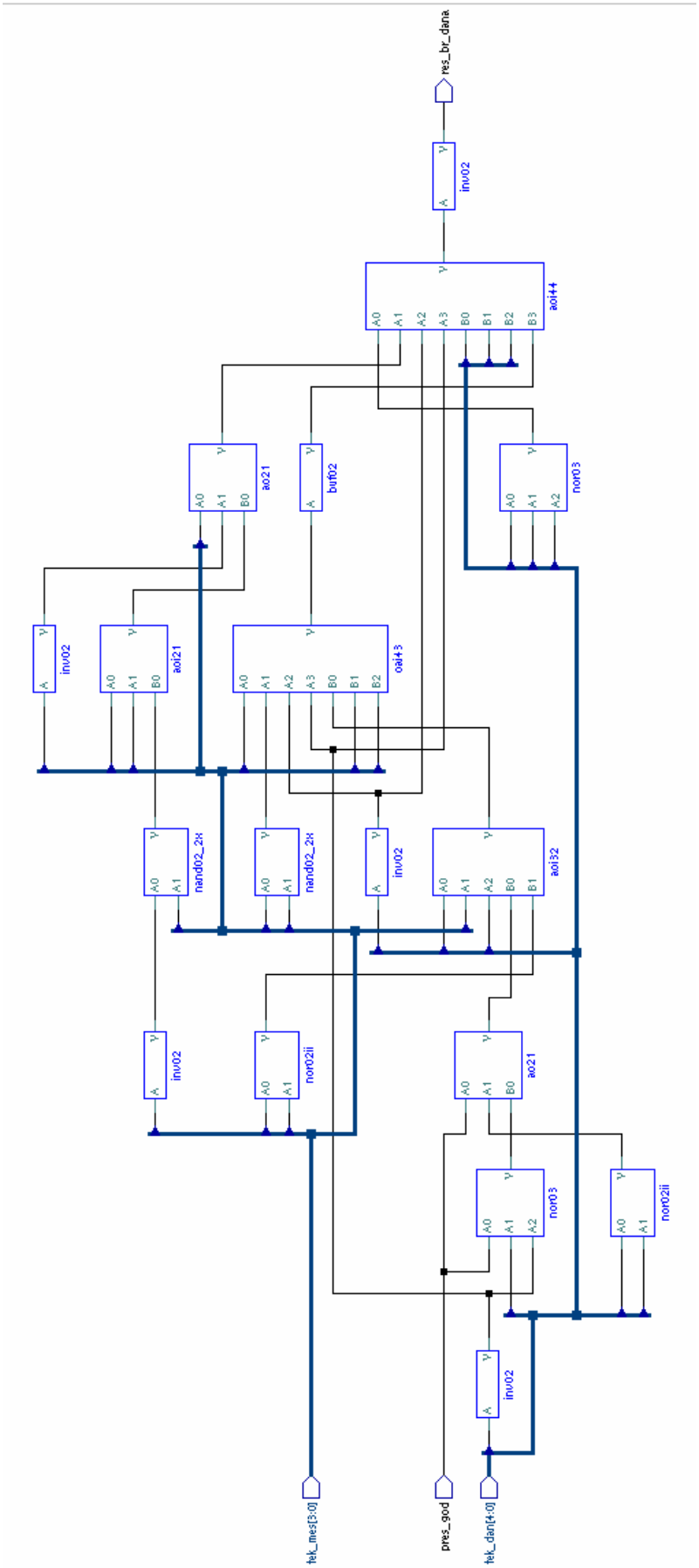
Šema sintetizovanog kola brojača godina:



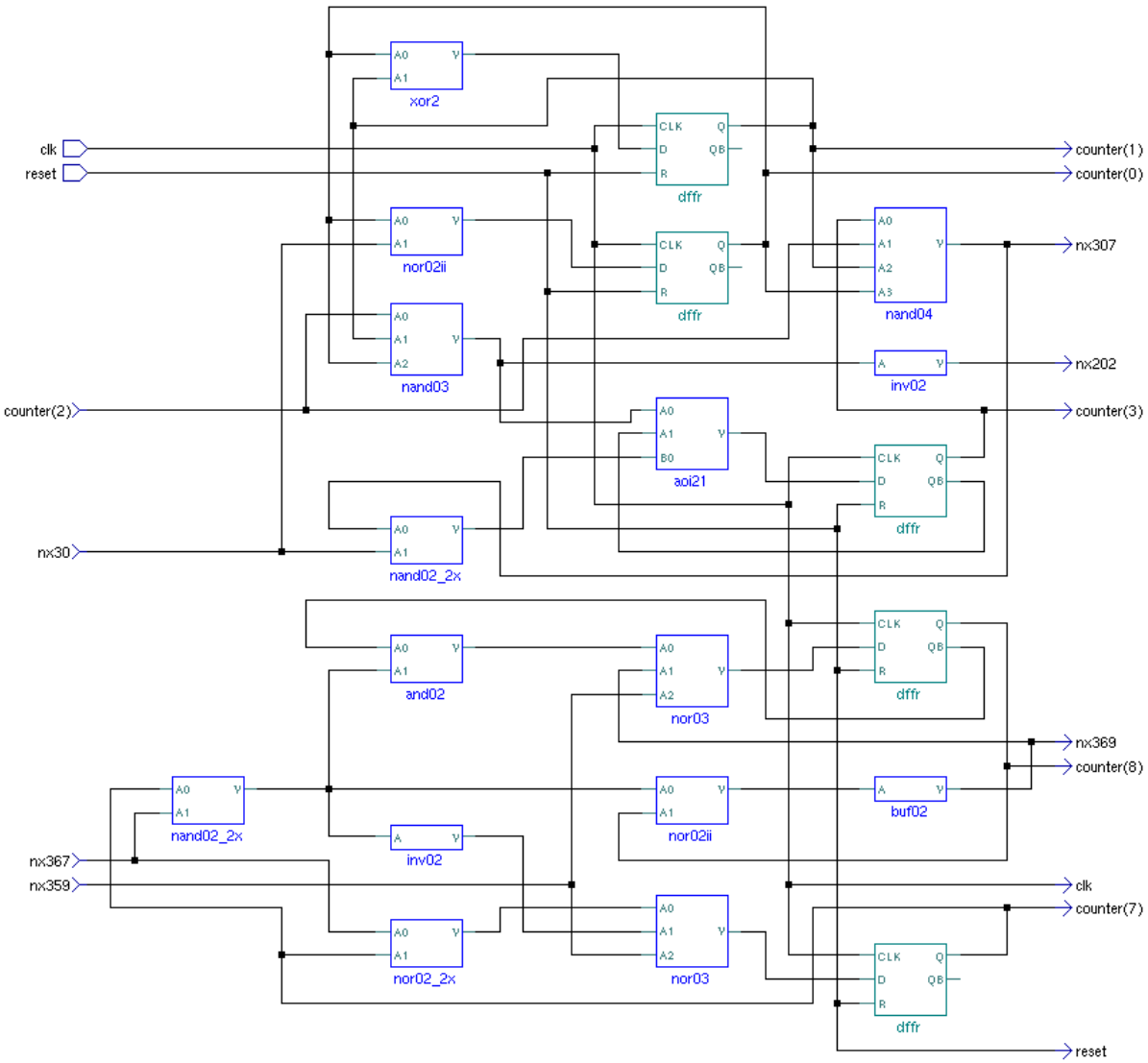
Šema sintetizovanog kola delitelja sa 4:



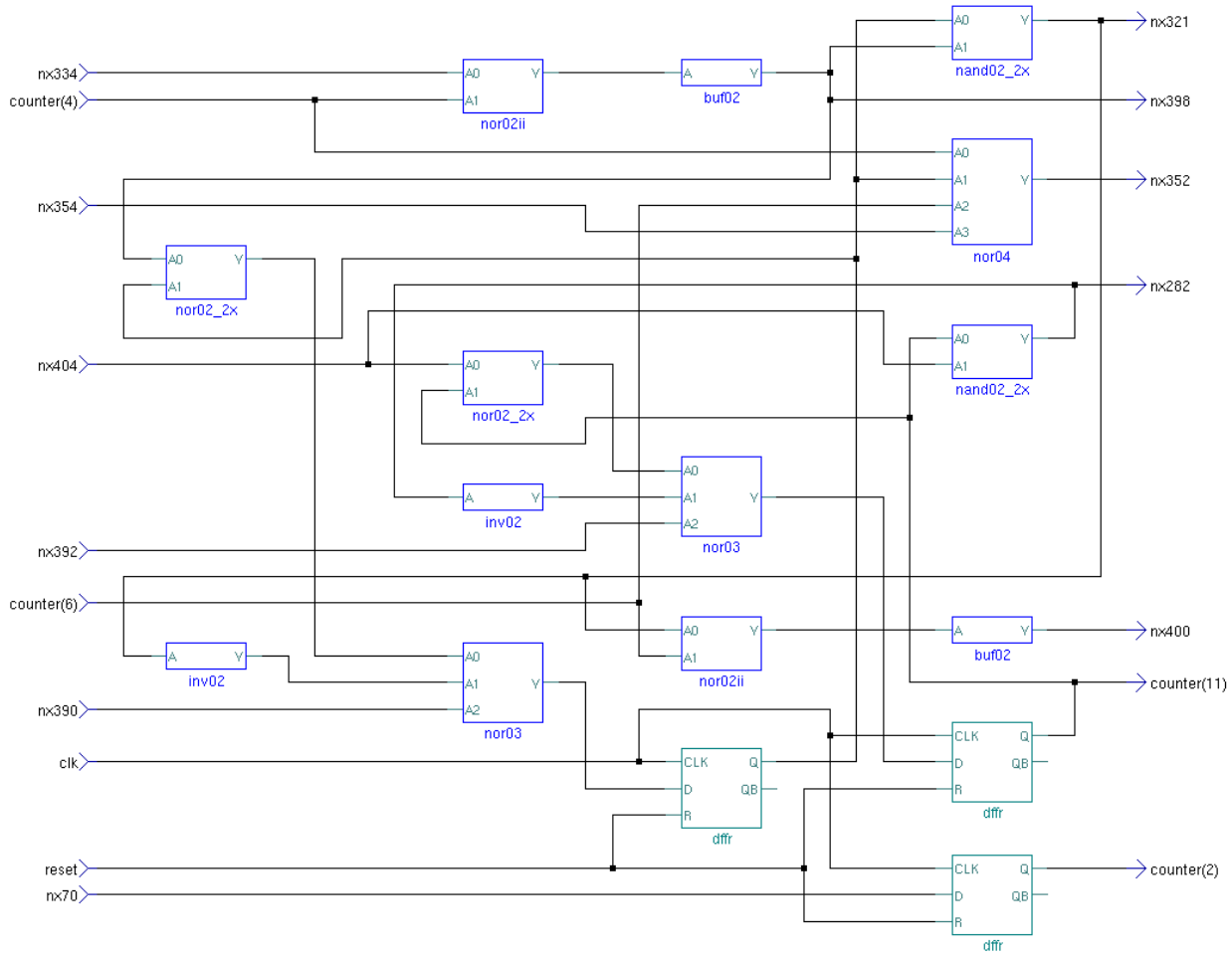
Šema sintetizovanog kola za reset brojača dana:



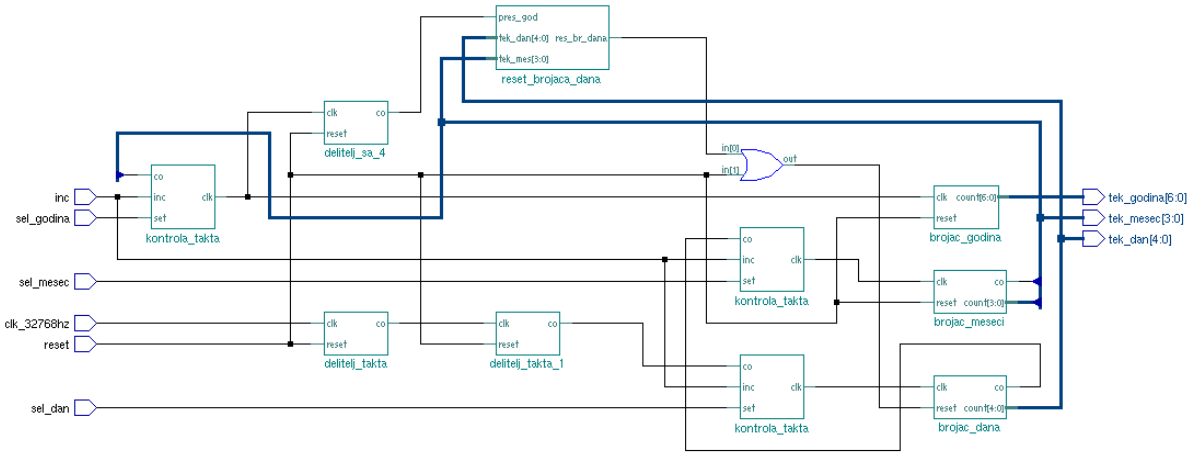
Šema sintetizovanog kola delitelja takta:



Šema sintetizovanog kola delitelja takta 1:

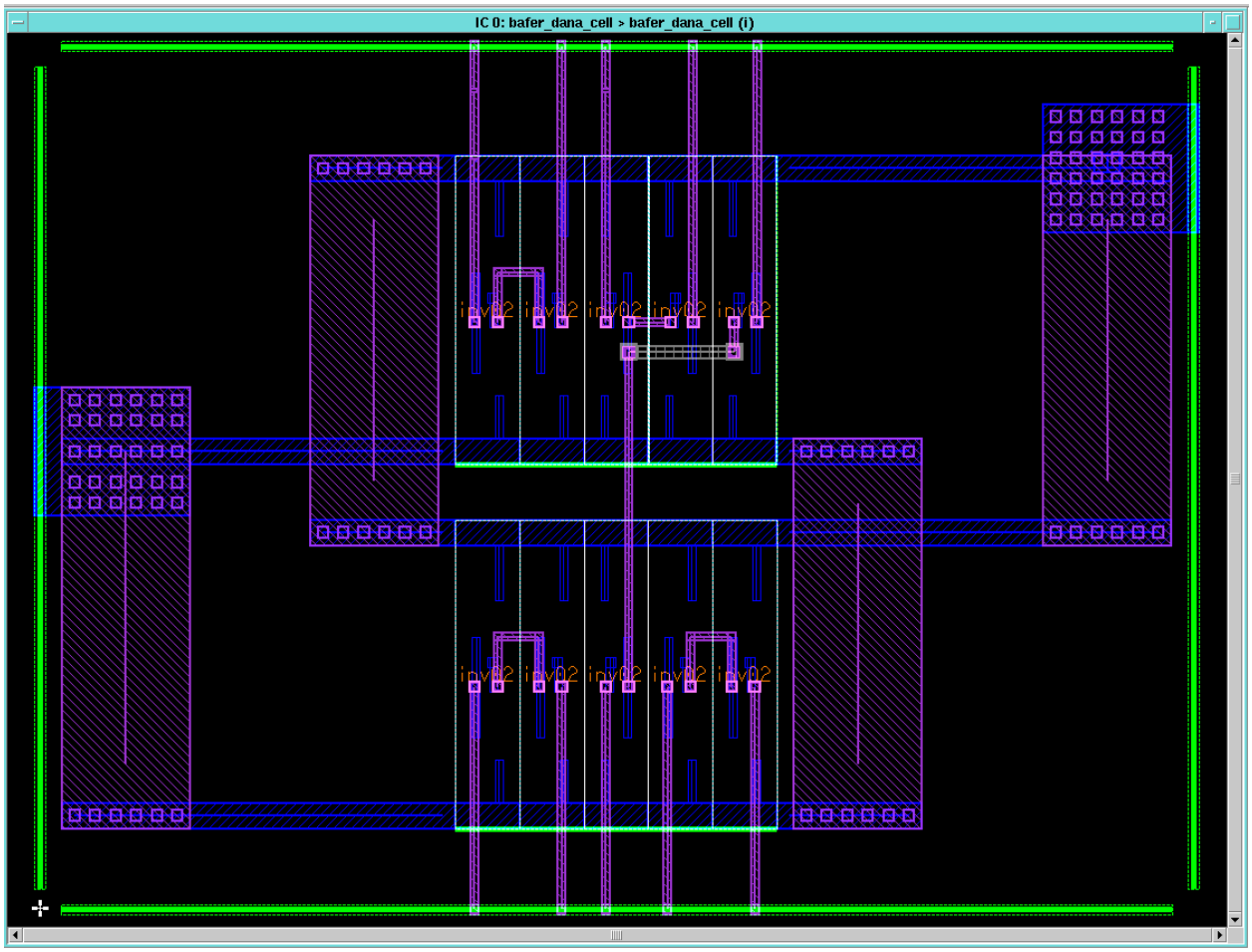


Šema komletnog kola:

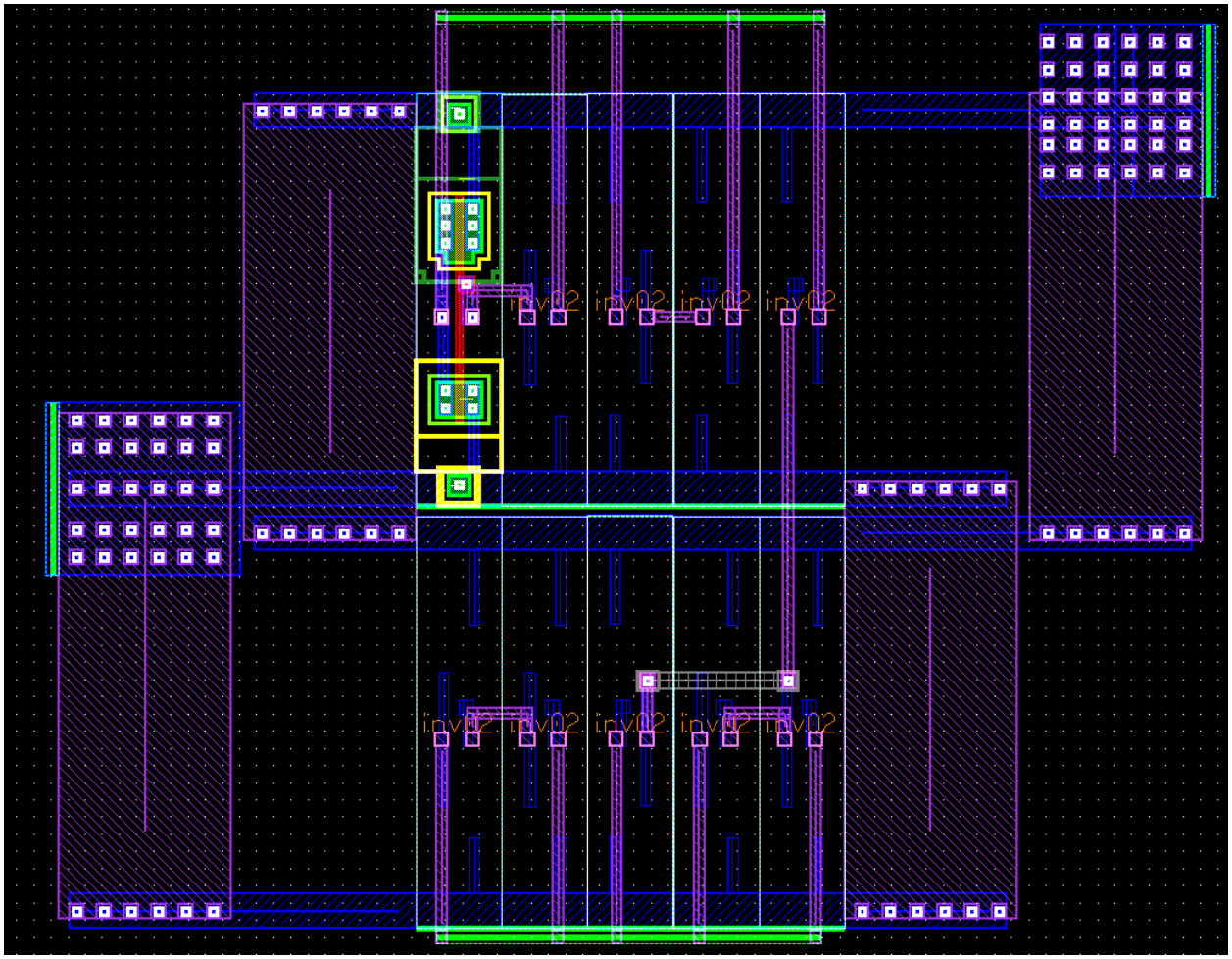


9. Prilog C. Prikaz layout-a makro-ćelija

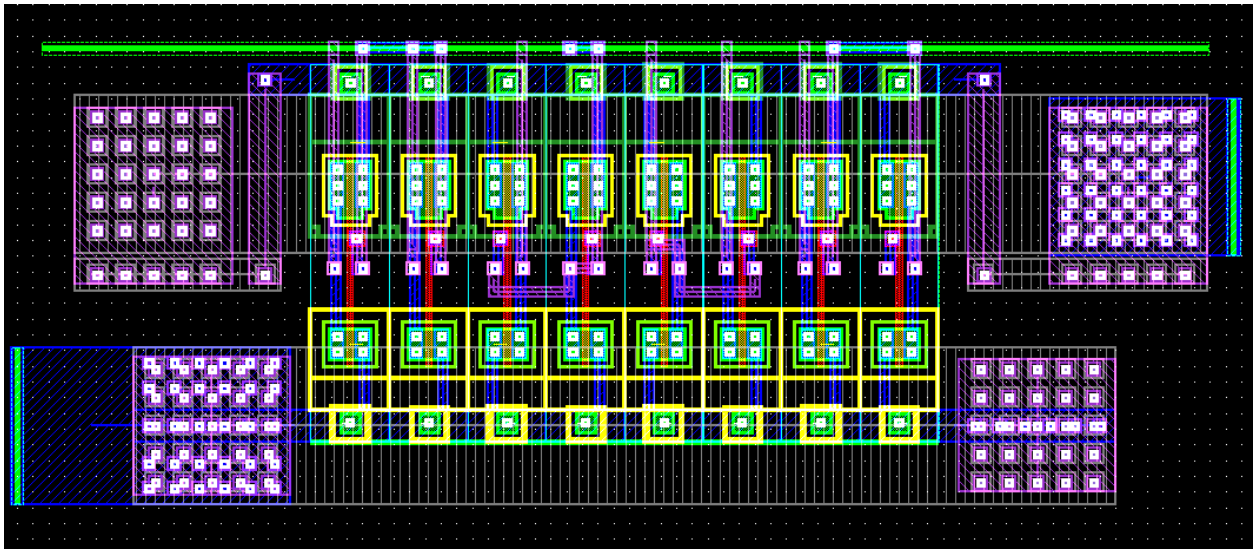
layout bafera dana:



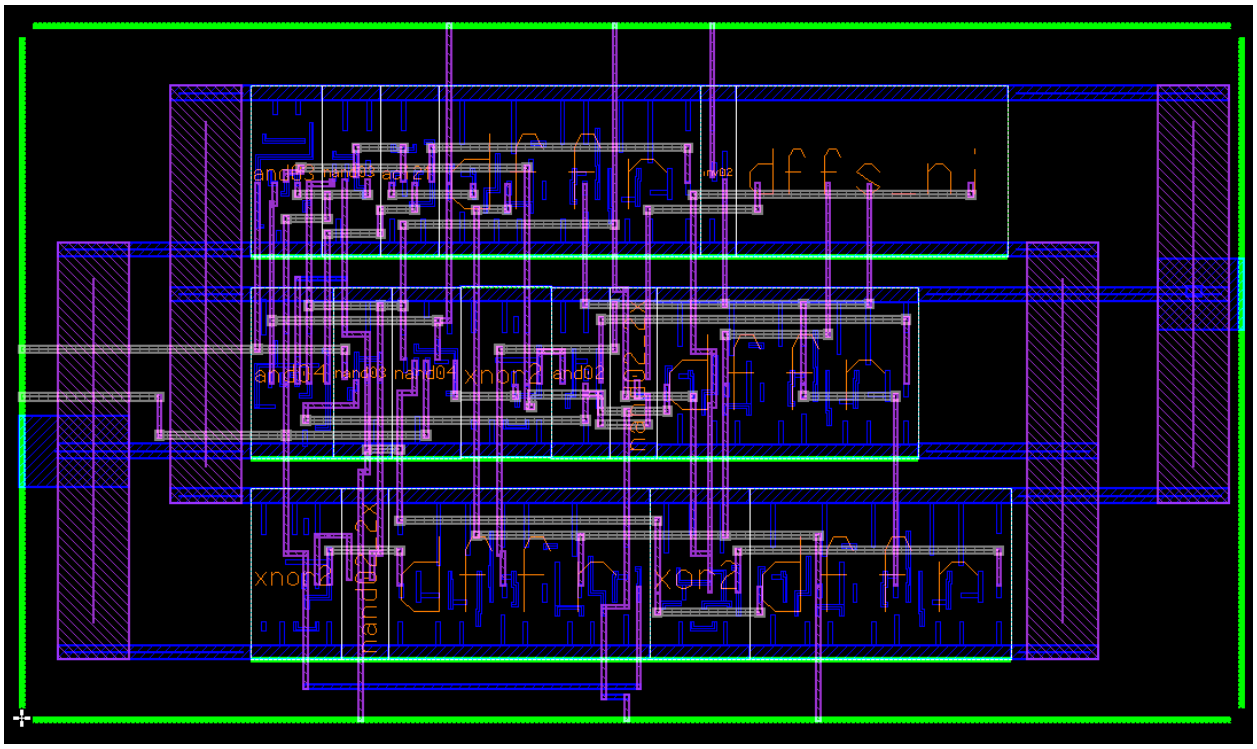
Layout bafera dana u kompaktnoj formi sa prikazanim layer-ima CMOS para



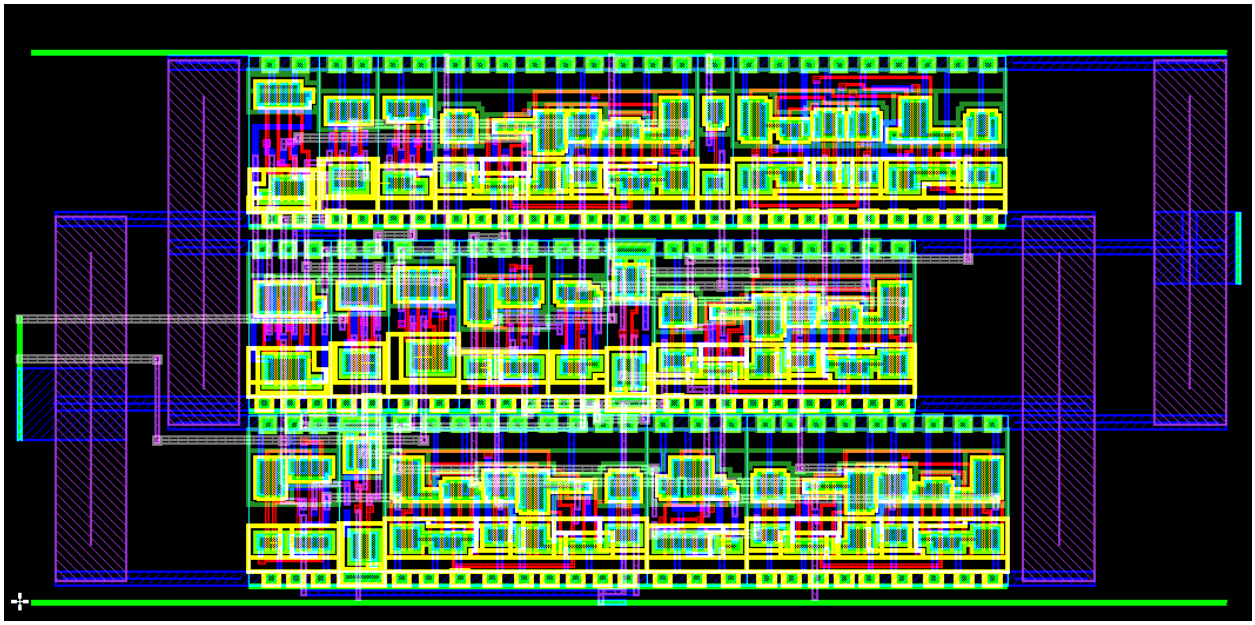
Layout bafera meseci:



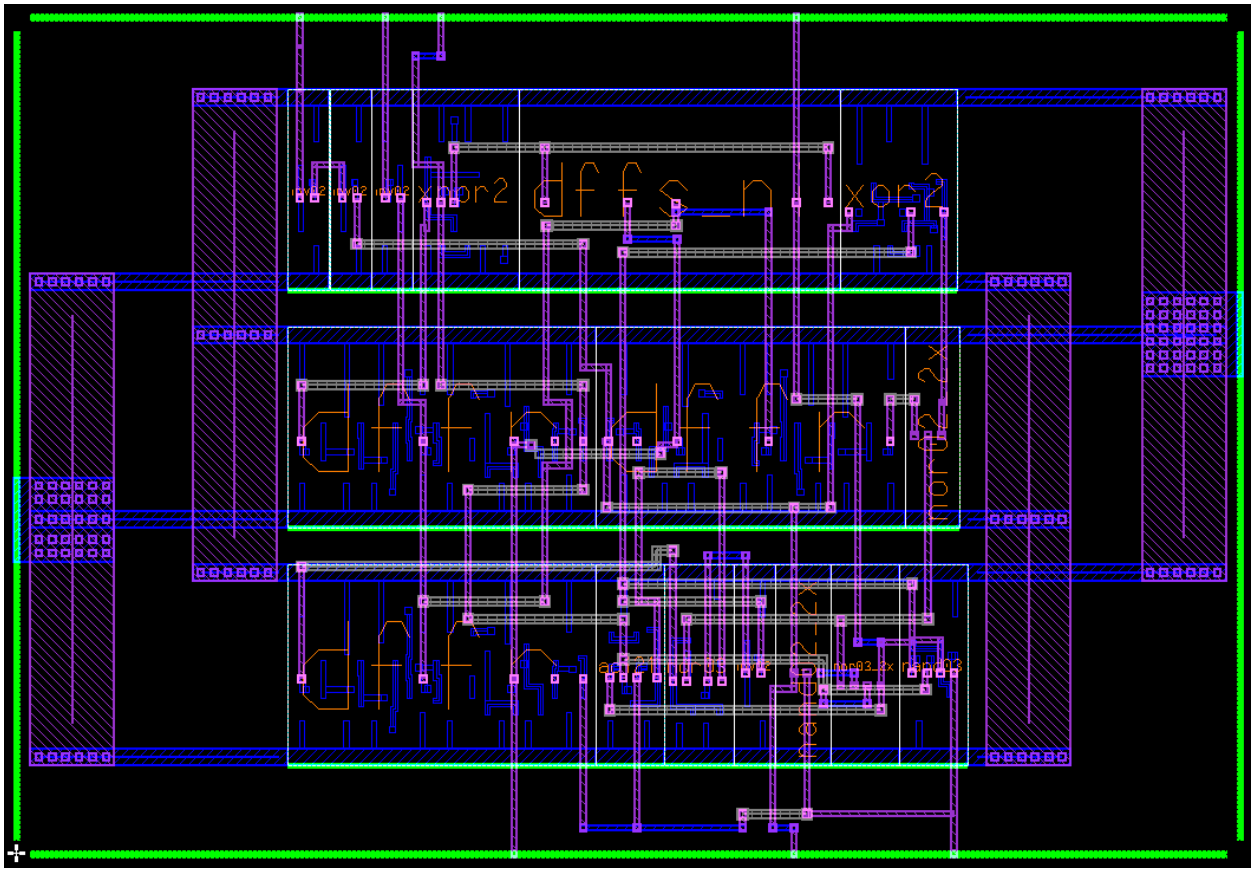
Layout brojča dana:



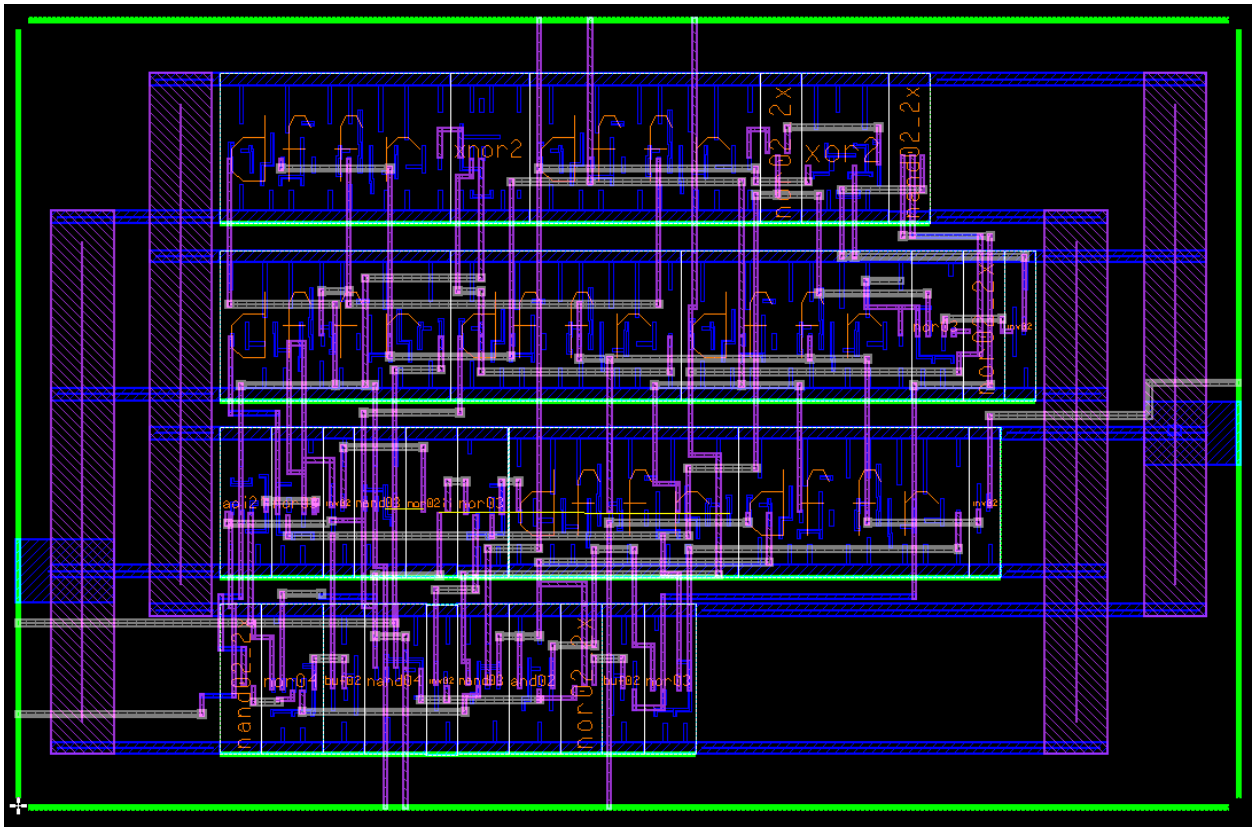
Layout brojača dana sa prikazom svih nivoa:



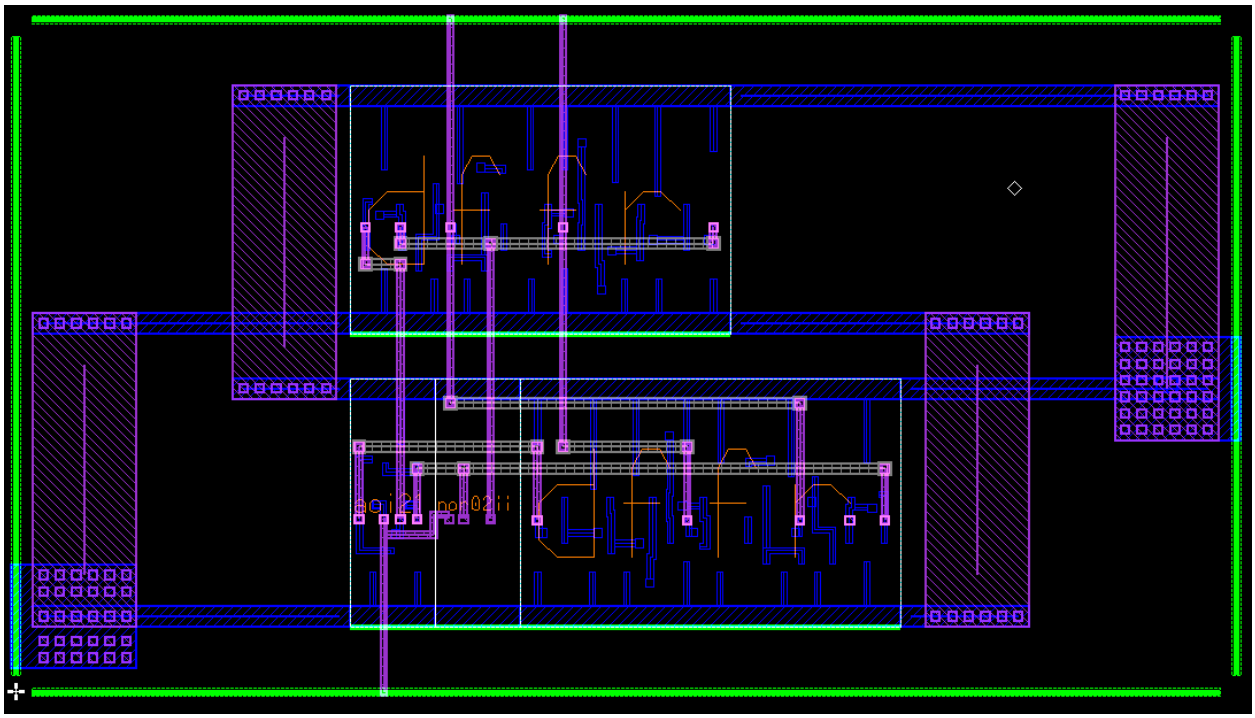
Layout brojača meseci:



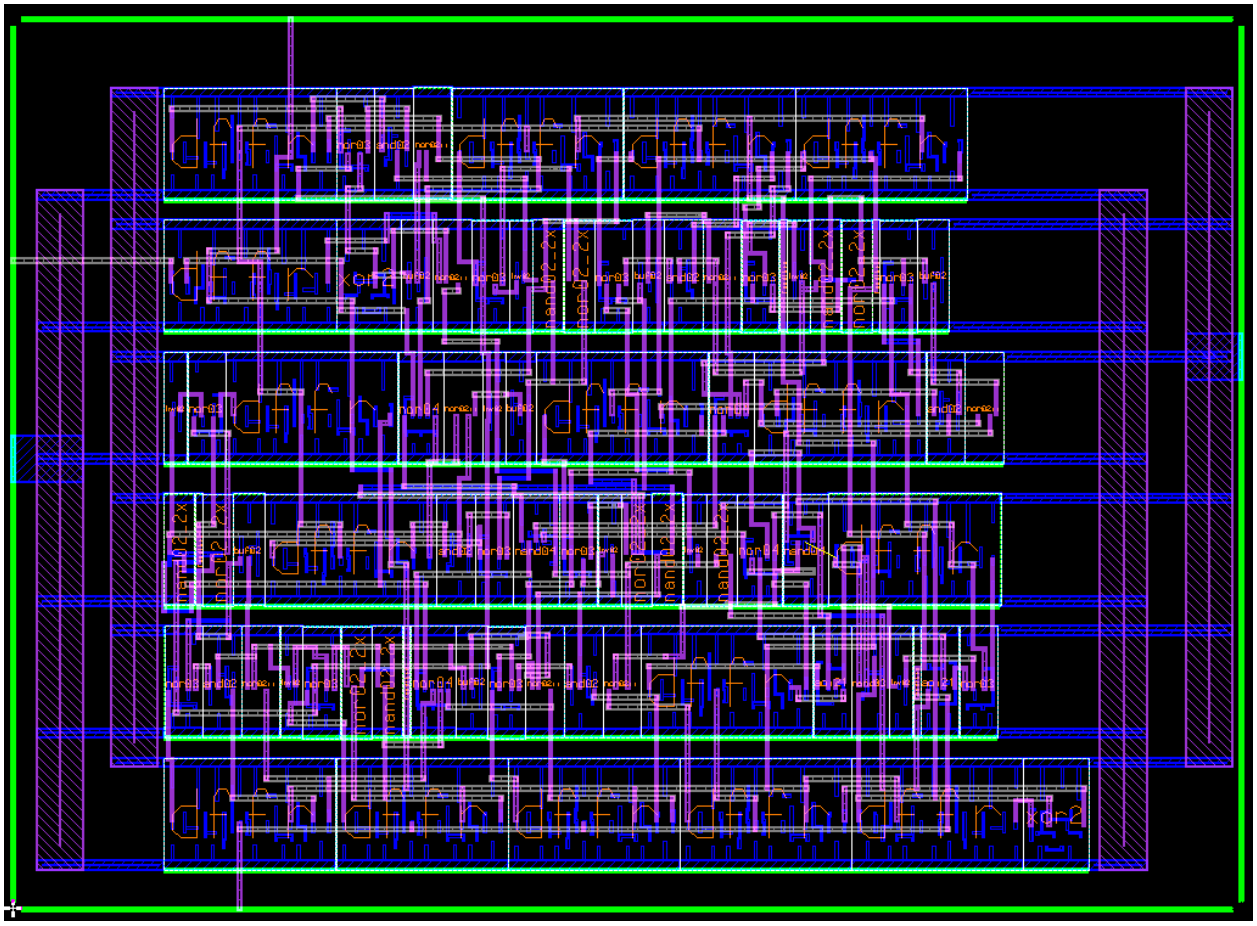
Layout brojača godina:



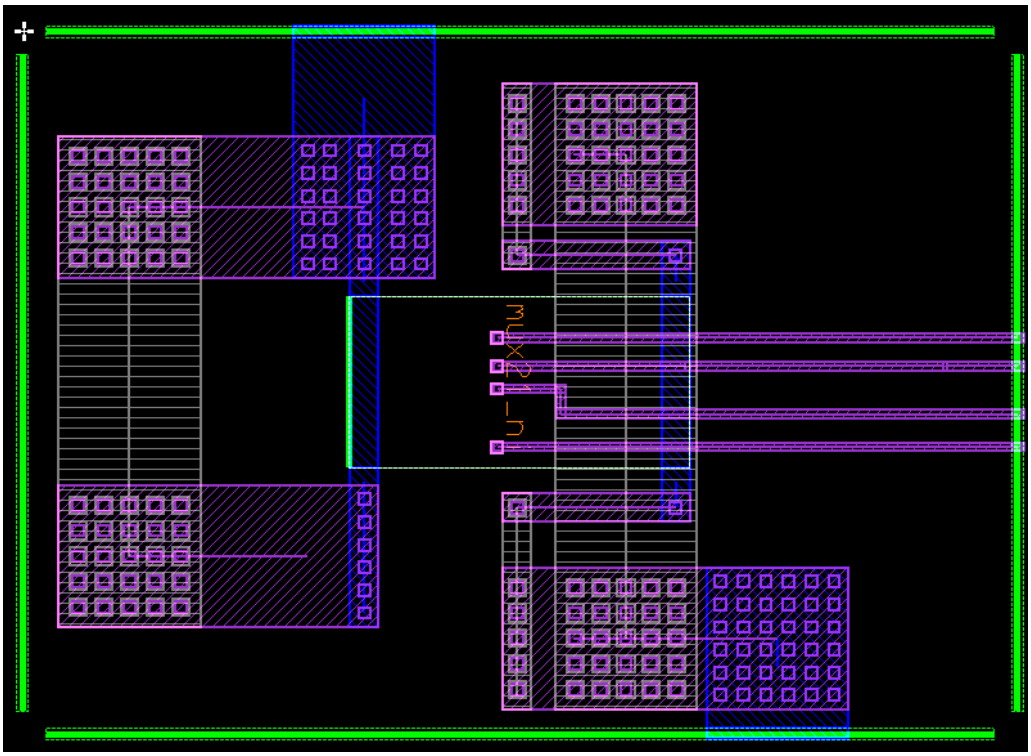
Layout delitelja sa 4:



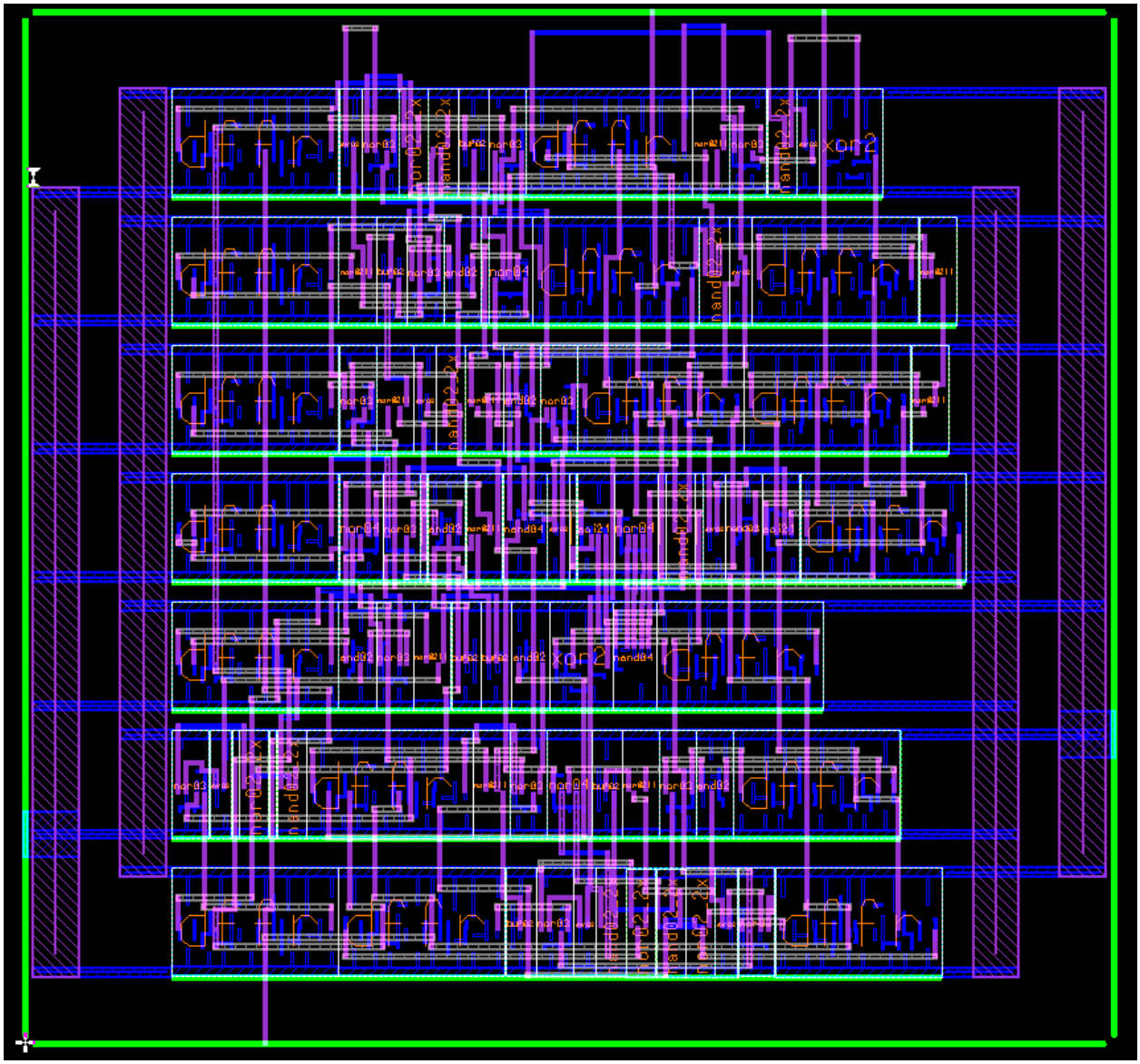
Layout delitelja takta:



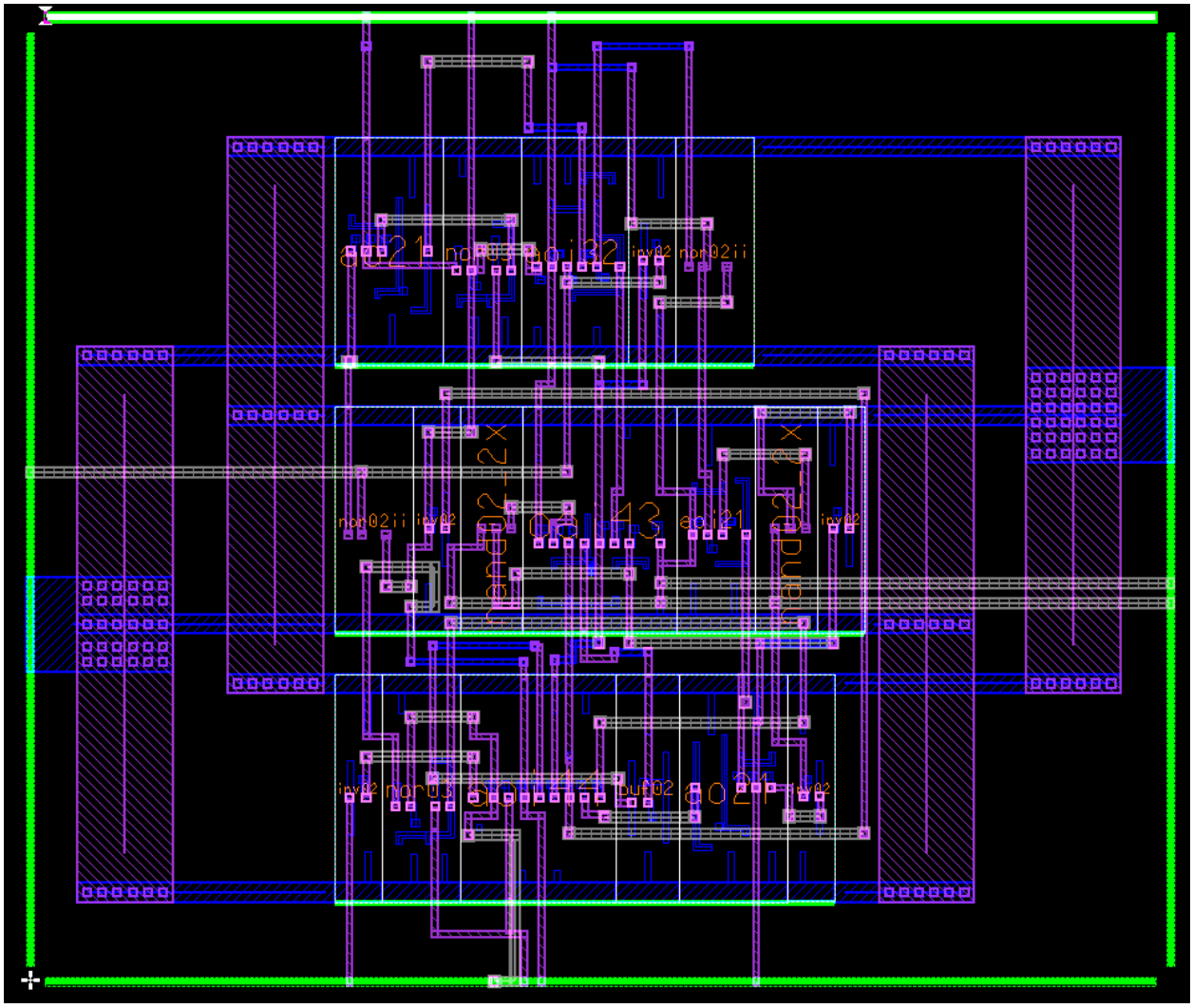
Layout modula za kontrolu takta:



Layout delitelja takta 1:

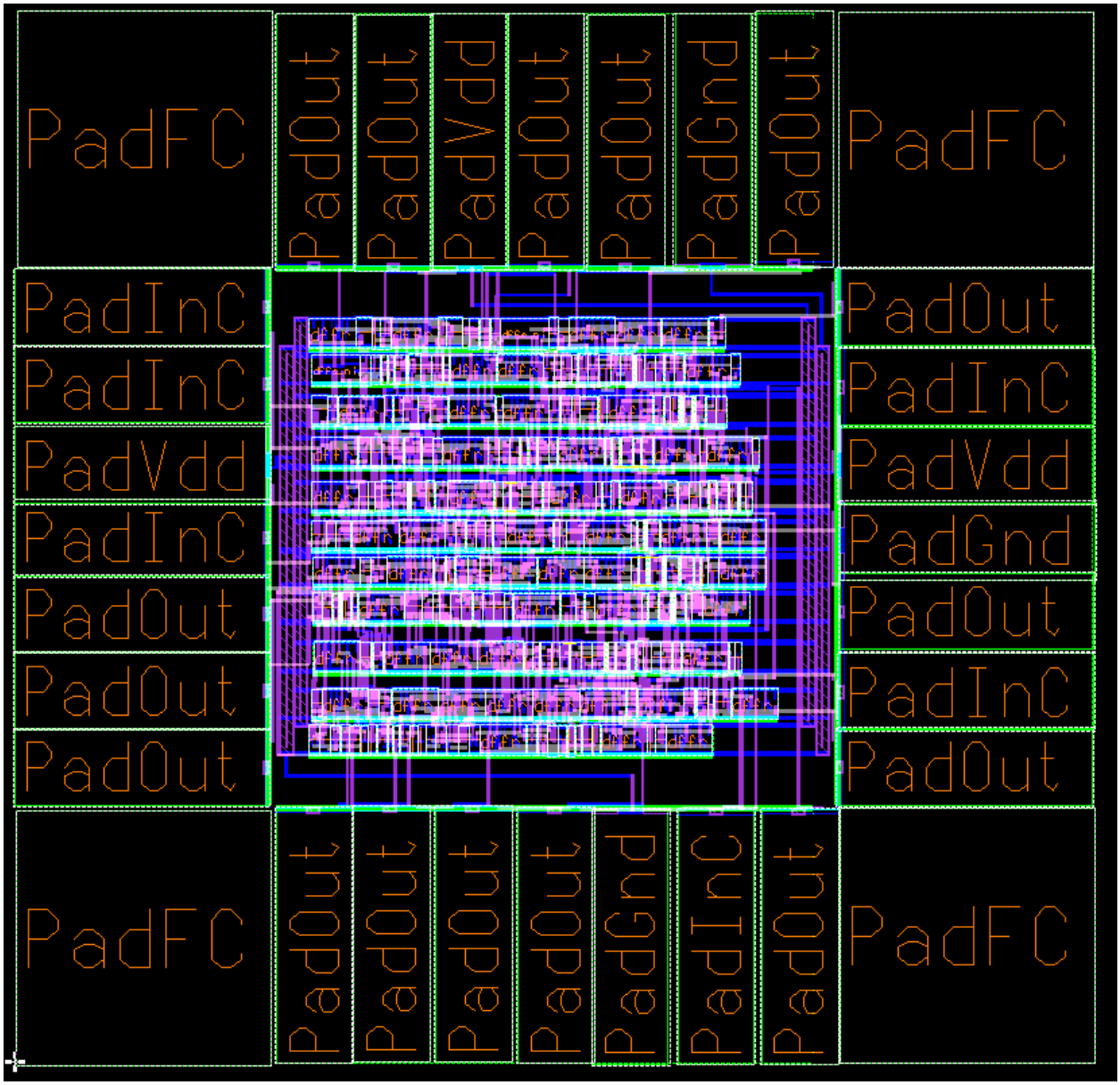


Layout modula za restartovanje brojača dana:

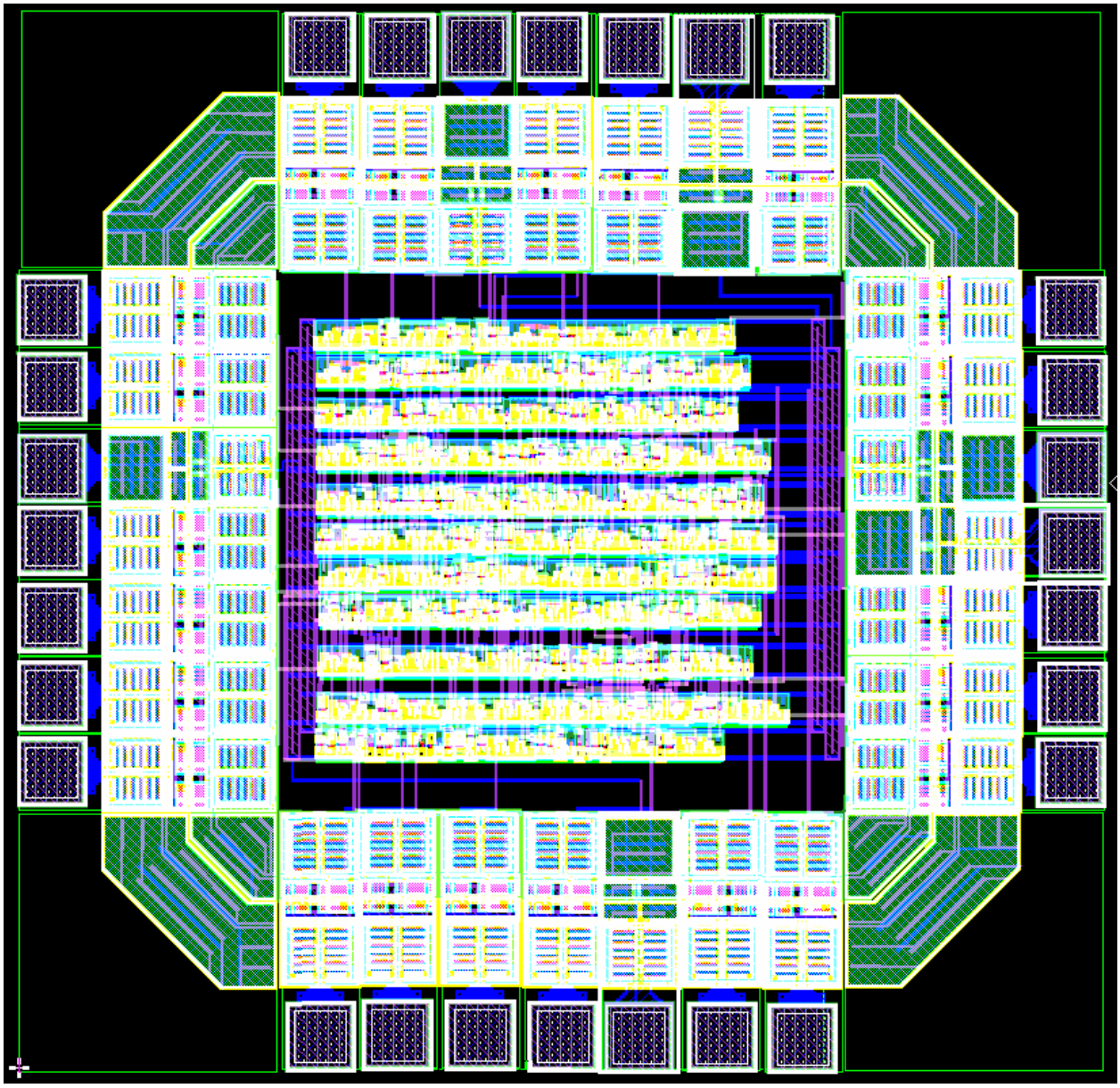


10. Prilog C. Prikaz lejauta čipa

Layout kompletnog čipa kalendara sa stopicama:



Layout kompletnog čipa kalendara (prikaz svih layer-a)



Sadržaj: