Glava 10

PROGRAMIRANJE PLD

Da bi se programabilne logičke komponente programirale, odnosno da bi im se ugradila željena funkcija, neophodno je korišćenje odgovarajućih softverskih alata. Pri tome, treba imati u vidu da postoje dva ključna koraka u projektovanju PLD. Prvi je vezan za projektanta i njegovu viziju projekta. On opisuje šta kolo treba da radi. Na osnovu toga softver sintetizuje logičku strukturu (logička šema) kojom može da se realizuje taj zadatak. Ovaj korak predstavlja strukturnu fazu projektovanja – kao što je rečeno u uvodnom poglavlju. On je, u najvećoj meri, tehnološki nezavisan. Prenos projekta na hardver predstavlja drugi korak, koji se naziva fizičko projektovanje. Za njegovu realizaciju koriste se softverski alati koji su veoma zavisni od tehnologije, odnosno vrste i tipa PLD koju treba programirati. Iako postoje integrisani alati koji podržavaju obe faze projektovanja i dalje se, u praksi, za strukturno i fizičko projektovanje koriste različiti alati. Šta više, oni su nezavisno razvijani i kasnije integrisani u vidu primene dva odvojena modula. Pri tome, alat za strukturno projektovanje na izlazu generiše podatke u formatu koji može da prihvati alat za fizičko projektovanje. Najčešće su to formati *Verilog* ili VHDL jezika za opis hardvera.

Strukturno projektovanje, odnosno sinteza, predstavlja kreativniji deo procesa prenosa ideje na hardver. Zahteva veću ulogu projektanta. Ona se ispoljava, najpre, kroz izbor tehnike ili metoda projektovanja. Nedavno je u [Tay16a] dat pregled deset najčešće korišćenih metoda projektovanja PLD. Jasno je, samo na osnovu njihovog broja, da je ova knjiga „tesna“ da bi obuhvatila objašnjenje svakog od njih. Zato samo upućujemo znatiželjne

čitaoce da pogledaju ovu referencu kako bi stekli bolji uvid u složenost posla projektanta.

Dobro je znati da isti problem može da se reši primenom različitih tehnika projektovanja. Izbor optimalnog metoda zavisi od cilja projekta kao i od veština kojima projektant raspolaže. Projektanti koji raspolažu sa više veština iz ove oblasti lakše će naći rešenje za konkretan problem. Pored izbora metoda projektovanja, projektant mora da zna da opiše zahteve na jeziku koji je razumljiv računaru da bi on (računar) našao rešenje koje ispunjava zadate zahteve. I za ovu svrhu postoji više rešenja. Pored više decenija starih jezika za opis hardvera, **HDL** (*Hardware Description Language*) tipa VHDL i *Verilog*, u novije vreme razvijeni su alati koji olakšavaju projektovanje na sistemskom nivou, **HLS** (*High Level Synthesis*). Uprkos tome, HDL pristup ne gubi na značaju, tako da se VHDL i *Verilog* nalaze u osnovi najznačajnijih alata za projektovanje PLD. Ubedljivo najveći broj elektronskih kola koja se danas koriste projektovana su primenom jednog od ova dva jezika. Dobar pregled HDL i HLS alata dat je u [Tay16b]. U primeru koji sledi mi ćemo se držati VHDL jezika (videti Glavu 3).

Softverski alati za sintezu najčešće podržavaju više tehnika projektovanja i više načina za opis projekta, a od projektanta se očekuje da izabere neku od ponuđenih opcija. Nezavisno od izabrane opcije, rezultat sinteze je logička šema razložena na blokove (logičke ćelije: NI, NILI, flip–flop,...) prepoznatljive za PLD koji se programira. Važno je istaći da su na ovom nivou razvoja projekta svi blokovi definisani svojim logičkim funkcijama i internim dinamičkim parametrima (vremena uspostavljanja i kašnjenja signala), a njihove fizičke dimenzije nisu uzete u obzir. Istovremeno, tačna dužina veza nije poznata, tako da se ne zna koliko je kašnjenje na vezama. Svi ovi podaci sadržani su, najčešće, u *Verilog* formatu. Napominjemo da se ova datoteka automatski generiše nezavisno od toga da li je projekat opisan u VHDL, *Verilog* ili nekom od HLS jezika. Zato nije neophodno (ali je korisno) da projektant zna sintaksu *Verilog* jezika.

Na osnovu fajla koji generiše alat za sintezu, odnosno poznate logičke šeme (opisane u *Verilog* formatu), alat za fizičko projektovanje obavlja sledeće zadatke:

- prevodi logičku šemu u strukturu ćelija (blokova) koje fizički postoje na PLD,
- određuje optimalni razmeštaj blokova sa stanovišta dužine veza i/ili površine (broj zauzetih blokova),

- trasira potrebne veze između blokova i
- prosleđuje programatoru informaciju o tome koje prekidače unutar PLD treba aktivirati.

Proizvođači PLD nude odgovarajuće alate za projektovanje. Dominantan uticaj na tržištu FPGA imaju Xilinx www.xilinx.com, *Intel Corporation* http://www.intel.com koja je kupila kompaniju *Altera* www.altera.com, *Achronix Semiconductor* www.achronix.com, *Atmel* www.atmel.com, and *Cypress Semiconductor* www.cypress.com. Da bi se stekla slika o stanju na tržištu softvera za programiranje PLD, pomenućemo samo neke od alata koje distribuiraju neki od najvećih proizvođača FPGA:

- *Xilix: ISE Design sSuite, Vivaldo Design Suite*,
- *Intel* (*Altera*): *Quartus® Prime design software* i *ModelSim–Altera Software*, verzija *ModelSim®* za *Altera* PLD,
- *Achronix Semiconductor: ACE Design Tool*,
- *Atmel: Atmel Studio IDE.*

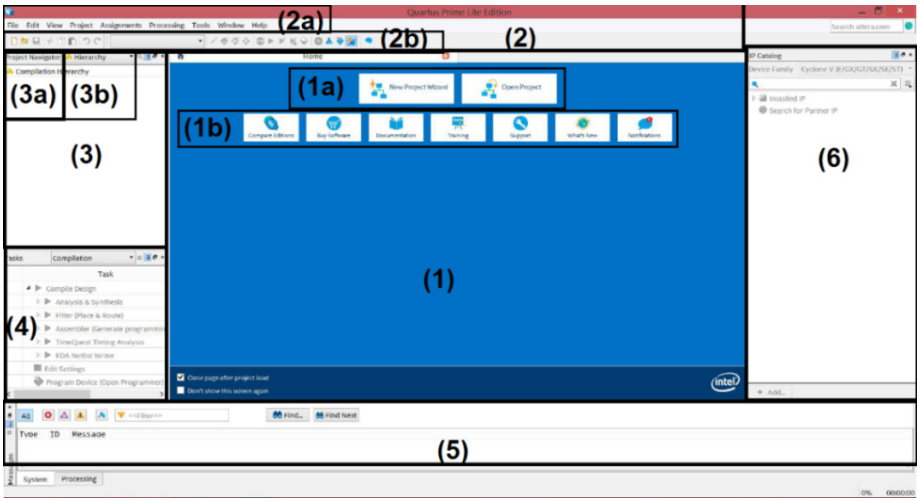
U daljem tekstu biće objašnjena primena alata *Quartus* na konkretnom primeru. U ovom poglavlju ćemo na jednom, relativno jednostavnom, primeru proći sve korake neophodne za realizaciju projekta na FPGA iz familije *Cyclon IV*.

10.1. *Quartus Prime*© 16.1

Realizovaćemo generator 32–bitnih pseudoslučajnih brojeva zasnovan na primeni algoritma **MWC RNG**, (***M**ultiply–**W**ith–**C**arry **R**andom **N**umber **G**enerator*) [Mar03]. Nećemo ulaziti u suštinu matematičkog problema, nego ćemo, kao početni korak u projektovanju, koristiti VHDL opis generatora na funkcionalnom nivou iz [Miš08].

10.1.1. Otvaranje novog projekta

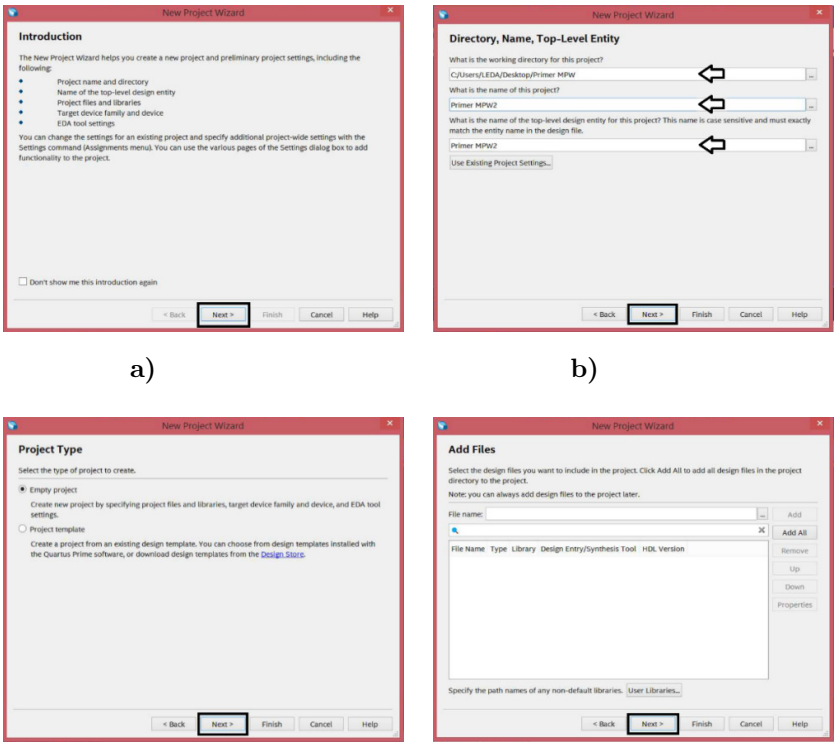
Po pokretanju programa *Quartus* 16.1, otvara se prozor prikazan na sl. 10.1. Uočava se šest celina. Najveći je centralni – radni prozor (1). Na početku, u njemu se nude prečice do osnovnih radnji za pokretanje projekta



Slika 10.1. Početni prozor pri otvaranju programa *Quartus* 16.1

New Project Wizard i *Open Project* (1a), kao i pomoć za obuku u rukovanju alatom (1b). Na vrhu prozora, horizontalno, nalazi se lista komandi ispod kojih su smešteni padajući meniji (2a), što je karakteristika većine programa koji rade pod *Windows* operativnim sistemom. Odmah ispod, nalaze se ikonice preko kojih se direktno aktiviraju neke od najčešće korišćenih naredbi (2b). Sa leve strane nalazi se prozor za „upravljanje“ kretanjem kroz projekat, *Project navigator* (3a). U istom prozoru smeštena je i kartica (tab) kojom se aktivira hijerarhijski prikaz projekta, *Hierachy* (3b). Ispod njega je prozor *Tasks* (4) u kome je smeštena lista zadataka koje treba uraditi da bi se realizovao projekat. U dnu ekrana nalazi se prozor u kome se prikazuju poruke o toku realizacije projekta *Messages* (5). Sa desne strane je prozor u kome se prikazuju raspoloživi resursi (biblioteke) za realizaciju projekta na konkretnom tipu FPGA, *IP Catalog* (6).

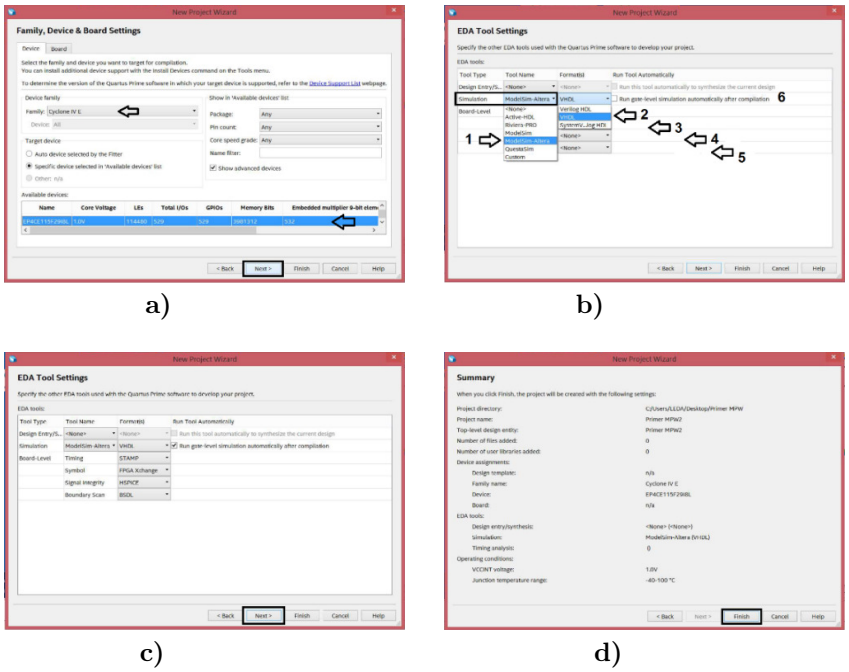
Aktiviranjem dugmeta *New Project Wizard* iz polja (1a) otvara se pomoćni program za definisanje novog projekta. Isti efekat ima i izbor opcije *New Project Wizard* iz padajućeg menija *File* u polju (2a). Nezavisno od načina na koji je ovaj program aktiviran, otvoriće se novi prozor, prikazan na sl. 10.2a, sa početnim informacijama o nameni programa. U njemu treba definisati naziv i lokaciju (direktorijum) novog projekta, naziv glavnog entiteta (na najvišem hijerarhijskom nivou), potrebne biblioteke i fajlove, tip i model FPGA na kome će se realizovati projekat i alate koji će se koristiti za opis, sintezu i verifikaciju projekta. Od projektanta se očekuje samo da pritisne dugnme *Next* u dnu ekrana. Otvoriće se ekran za unos



Slika 10.2. Prva četiri koraka u definisanju parametara novog projekta

lokacije radnog direktorijuma (kod nas je to *C:/Users/LEDA/Desktop/Primer MPW*), naziva projekta (*Primer MPW*2) i naziv glavnog entiteta (*Primer MPW*2), sl.10.2b.

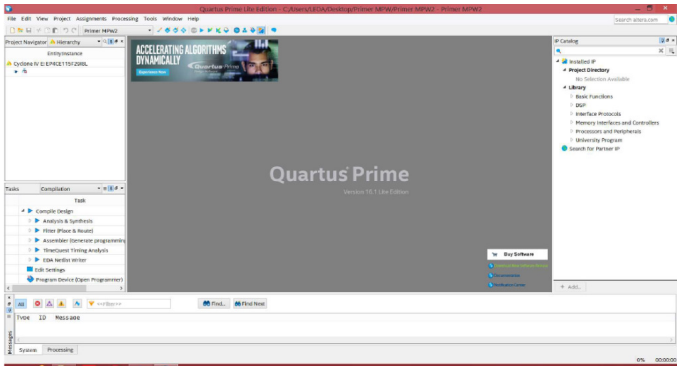
U narednom koraku bira se da li počinje potpuno novi, prazan (*empty*) projekat ili se koristi šablon već postojećeg projekta. Drugi slučaj omogućava da se preskoče naredni koraci. Mi biramo opciju *Empty Project*, kao na sl. 10.2c. S obzirom da se radi o prvom, i potpuno novom projektu, ne postoje fajlovi koje bismo mogli da koristimo, pa u narednom prozoru (sl. 10.2d) selektujemo *Next*. U nastavku treba podesiti parametre koji se odnose na tip i model FPGA koji će se programirati. Na sl. 10.3a izabran je model EP4CE115F29IL18 iz familije *Cyclon IV* E. Najzad, potrebno je podesiti softverske alate. Oni se biraju iz padajućih menija, kao što prikazuje sl. 10.3b, a konačan izbor svih alata ilustrovan je na sl. 10.3c. Na kraju, sledi prozor u kome su ispisane sve izabrane opcije. Projektant može da koriguje



Slika 10.3. Završni koraci u definisanju parametara novog projekta

unete podatke povratkom u odgovarajući prozor preko dugmeta *Back*, ili da završi postavljanje podataka izborom dugmeta *Finish*, kao što prikazuje sl. 10.3d.

Po završetku, javiće se u glavnom prozoru poruka „*Please wait while Quartus Prime opens project Primer MPW2*“. Kao konačni rezultat ove aktivnosti, otvoriće se prozor prikazan na sl. 10.4.

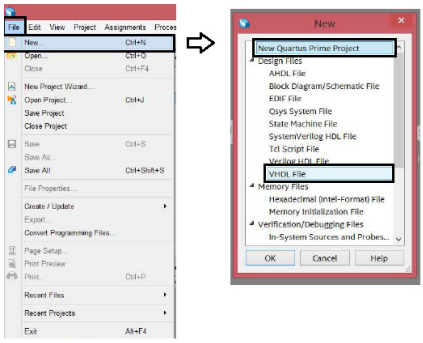


Slika 10.4. Izgled ekrana po otvaranju novog projekta

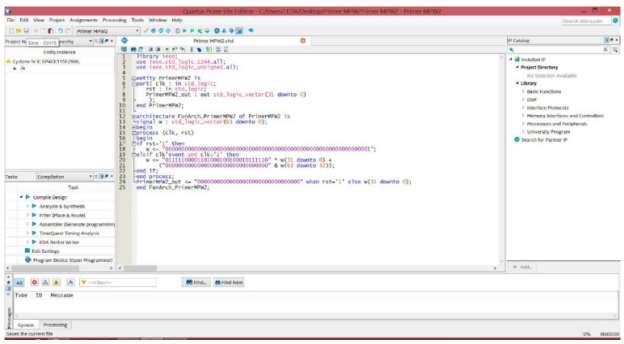
10.1.2. Unos VHDL opisa u *Quartus*

Da bi se počelo sa programiranjem FPGA, neophodno je opisati šta projekat treba da radi. S obzirom da je projekat potpuno prazan, treba otvoriti novi fajl. Zato se iz horizontalnog menija (2a) bira *File>New*. Time se otvara novi prozor namenjen za unos svih novih fajlova u projekat. Nas trenutno interesuje da unesemo opis MPW RG u formi VHDL kôda. Zato, iz ponuđenih opcija biramo *New Quartus Prime Project>Design Files> VHDL File*, kao što prikazuje sl. 10.5.

U glavnom prozoru otvara se VHDL editor sa ponuđenim nazivom dokumenta „Vhld1.vhd“ ispisanim iznad editora. Korisno je da se već u praznom prozoru definiše ime novog fajla izborom *File>Save As* upisom naziva projekta („Projekat MPW2“) u novom prozoru. Zatim treba uneti VHDL kod i zapamtiti izmenu (*File>Save*). Izgled ekrana po završenom VHDL opisu prikazan je na sl. 10.6.



Slika 10.5. Koraci neophodni za otvaranje VHDL editora



Slika 10.6. Izgled ekrana po završetku VHDL opisa

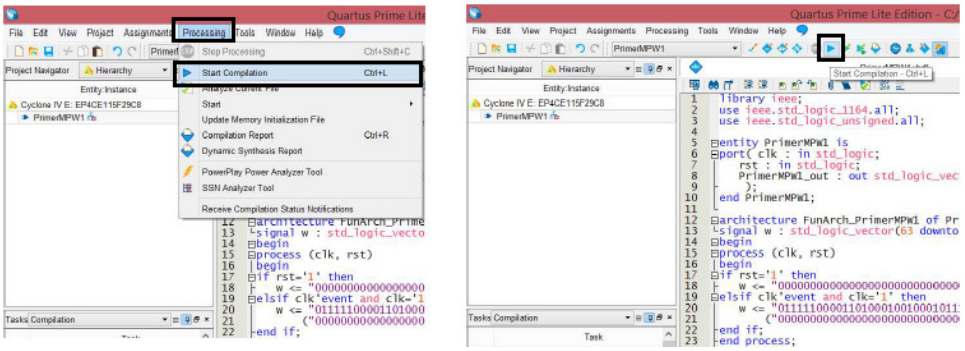
10.1.3. Prevođenje VHDL opisa

Sledeći korak predstavlja kompajliranje (prevođenje) VHDL kôda na logičke strukture koje fizički postoje u konkretnom FPGA kolu, odnosno automatska sinteza.

Ovaj proces može da se pokrene tako što se iz horizontalnog menija selektuje *Processing>Start Compilation* ili samo selektovanjem ikonice sa plavim trouglom. Oba postupka ilustrovana su na sl. 10.7a i 10.7b, redom.

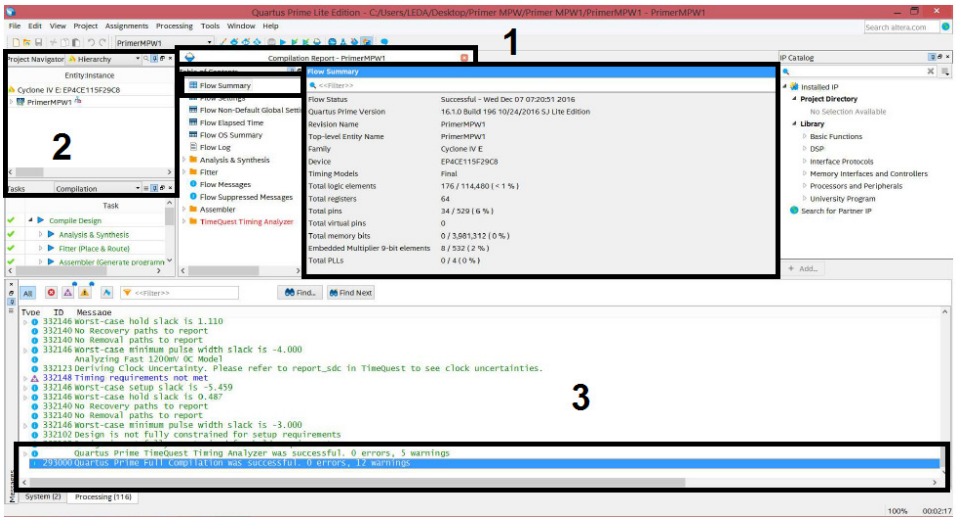
Proces prevođenja traje izvesno vreme, tako da za složene projekte zahteva strpljenje. Na kraju, u centralnom radnom prostoru dobije se izveštaj (*Compilation Report*), označen sa 1 na sl. 10.8, koji sadrži veliki broj informacija. Do njih se dolazi izborom odgovarajućeg polja u tabeli sa leve strane. Na sl. 10.8 prikazan je sadržaj konačnog izveštaja o toku kompilovanja (*Flow Summary*). On, osim podatka o uspešnosti, u prvom redu (*Flow Status Successful* datum), sadrži podatke o stepenu iskorišćenosti pojedinih logičkih celina u izabranom FPGA. U konkretnom primeru, s obzirom da je projekat relativno mali, a FPGA relativno veliki, zauzeto je samo 176 od 114,480 logičkih elemenata ili manje od 1%, kao i samo 34 od 529 pinova i 8 od 532 9–bitnih množača.

Po završetku prevođenja, u prozoru *Hierarchy,* prikazaće se kompletna hijerarhija upotrebljenih entiteta. U slučaju sa sl. 10.8 radi se samo o jednom hijerarhijskom nivou (ovaj deo izveštaja označen je na sl 10.8 sa 2).



a) b)

Slika 10.7. Pokretanje prevođenja VHDL koda u logičke strukture sa FPGA: a) preko padajućeg menija, b) prečica preko ikonice



Slika 10.8. Izgled prozora po završetku prevođenja

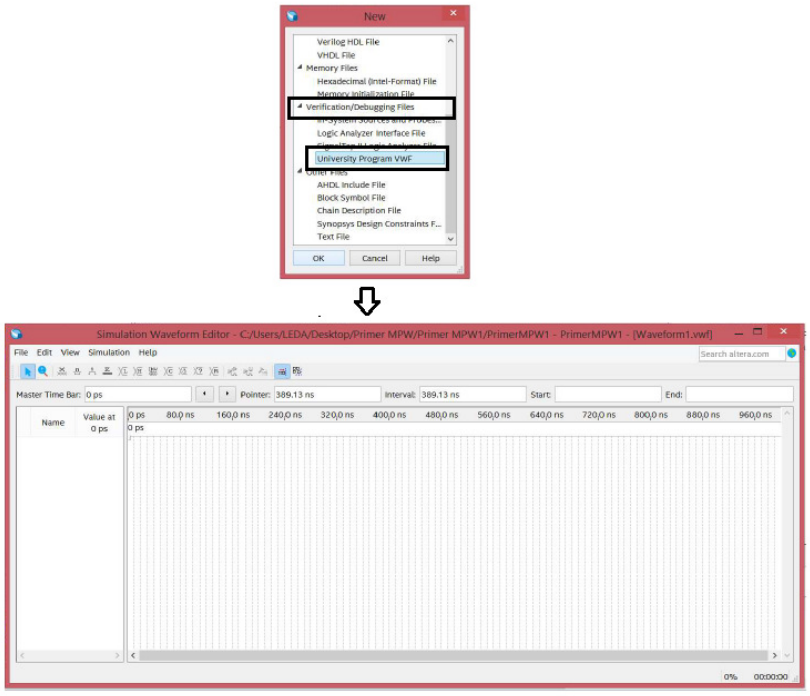
U dnu ekrana u polju *Messages* (označenom sa 3 na sl. 10.8) ispisuju se poruke o toku pojedinih koraka za vreme prevođenja. Uspešno završen proces propraćen je porukom sa konstatacijom da analizator kašnjenja nije otkrio greške: *Timing Analizer was successful* 0 *errors*. Rezultat je prihvatljiv i ukoliko se otkrije jedna ili više potencijalno kritičnih tačaka klasifikovanih kao upozorenja (*warnings*).

Po uspešno završenom prevođenju, poznati su podaci o dužini svih veza, a time i o kašnjenju na vezama. Zato je dobro da se pre prenošenja na hardver, projekat još jednom proveri simulacijom.

10.1.4. Verifikacija simulacijom

Kao što je navedeno u poglavlju 3, za verifikaciju se koristi *Test–bench* opcija iz VHDL editora. Da bi se verifikovao projekat, neophodno je osmisliti pobudne sekvence za testiranje (testni vektor) i njima pobuditi MPW RG. U okviru *Quartus* paketa postoji simulator koji se pobuđuje na sledeći način.

S obzirom da se radi o sasvim novoj aktivnosti u okviru projekta, pristupa joj se iz prozora *New*, do koga se dolazi selekcijom *File>New*. U njemu se, ispod polja *Verification/Debuging Files* izabere opcija, *University*



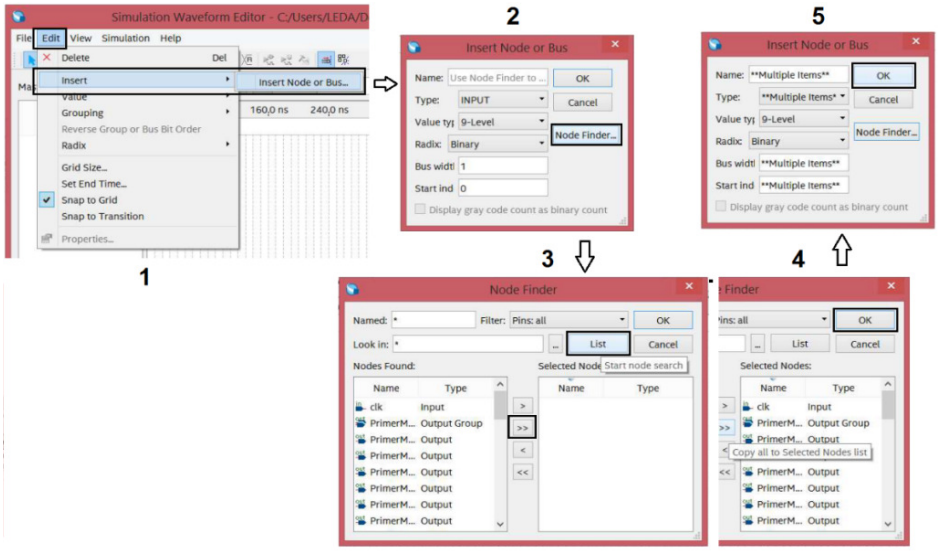
Slika 10.9. Otvaranje editora talasnih oblika (*Simulation Waveform Editor*)

Program VWF (sl. 10.9), kako bi se pokrenuo editor talasnih oblika u novom prozoru nazvanom *Simulation Waveform Editor* (u kome je reč *Vaweform* pogrešno napisana!), prikazan na donjem delu sl. 10.9. Komande su, u Windows maniru, raspoređene u dva reda na vrhu ekrana. U prvom su komande sa padajućim menijima, dok su u drugom ikonice. Ispod, u redu nazvanom *Master Time Bar*, nalaze se monitorski prozori u kojima se prati vremenski tok simulacije i položaj kursora. Levo dole je prozor u kome se nalazi spisak posmatranih signala, dok je centralni deo rezervisan za posmatranje talasnih oblika.

Pre početka simulacije treba uneti:

- čvorove čije talasne oblike želimo da posmatramo,
- vreme završetka simulacije i
- zadati pobudu, odnosno talasne oblike ulaznim signalima.

Unos signala koje želimo da posmatramo obavlja se u više koraka ilustrovanih na sl. 10.10.

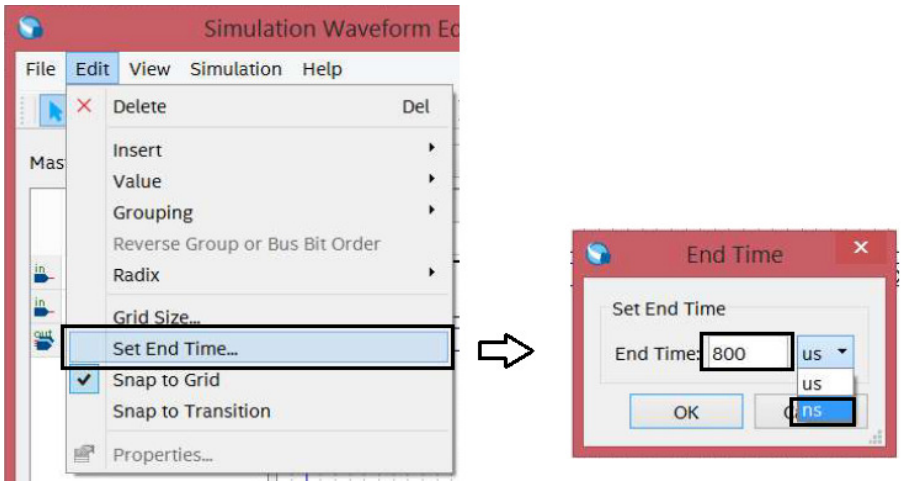


Slika 10.10. Selekcija čvorova čije talasne oblike želimo da posmatramo

- Najpre se selektuje *Edit>Insert>Insert Node or Bus*, [to je prikazano na sl. 10.10(1).
- Otvara se prozor *Insert Node or Bus*, sl. 10.10(2).
- Selektovanjem dugmeta *Node Finder*, otvara se prozor istog naziva, sl. 10.10(3). Kada se pritisne dugme List, u levom delu ekrana dobije se lista svih čvorova (definisanih kao signali u VHDL opisu).
- Korisnik može da selektuje pojedine čvorove i aktiviranjem dugmeta > da ih prosledi u desni deo ekrana koji prikazuje samo listu selektovanih čvorova. Izborom opcije >> selektuju se svi čvorovi, što je prikazano na sl.10.10(4).
- Lista se potvrđuje selekcijom polja OK čime se vraćamo u prozor Insert Node or Bus koji napuštamo, selekcijom polja OK, sl. 10.10(5).

Svi selektovani signali pojaviće se u levom delu glavnog prozora.

Po završetku specifikacije signala koje želimo da posmatramo, potrebno je definisati parametre i uslove simulacije. Najpre definišemo vreme završetka analize. Slekcijom *Edit>Set End Time*, u osnovnom simulacionom prozoru otvara se prozor *End Time* u koji se unosi brojna vrednost i merne jedinice. Sa sl. 10.11 vidi se da smo izabrali da simulacija traje do 800 ns.

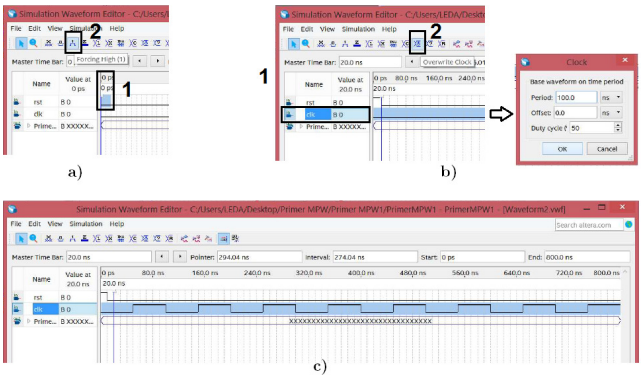


Slika 10.11. Postavljanje vremena završetka simulacije.

Ostalo je da se definišu pobudni signali. U ovom primeru postoje dva ulazna signala. To su reset signal (označen kao *rst*) i signal takta (označen kao *clk*). Dodeljivanje vrednosti obavlja se tako što se selektuje željeni signal iz spiska sa levog dela prozora, a onda mu se dodeljuje vrednost. U slučaju signala *rst*, želimo da bude u stanju 1 tokom prvih 20 ns, a zatim da bude na nuli. Ovo se postavlja definisanjem intervala kursorom uz pomoć miša, povlačenjem od 0 do 20 ns (dužina intervala prati se u prozorima na vrhu ekrana). Zatim se ide na ikonicu *Force High (1)*, čime se, za obeleženi vremenski interval, postavlja visoki logički nivo.

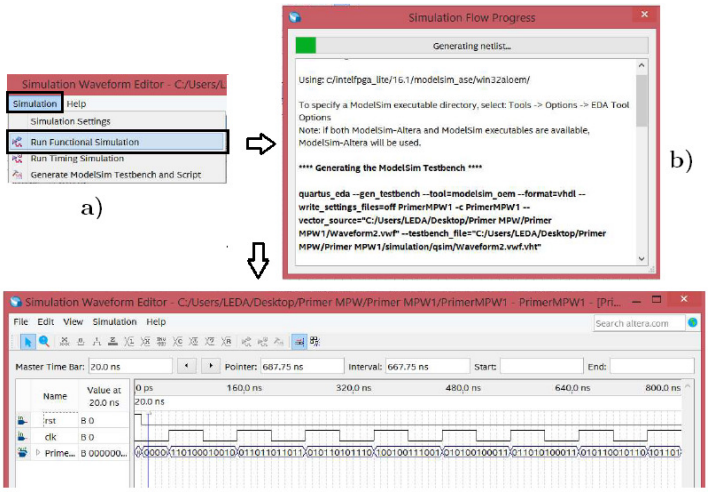
Zadavanje vrednosti signalu takta, koji periodično menja vrednost, nije racionalno na opisani način. Za ovu namenu predviđena je opcija direktnog zadavanja parametara periodičnih signala u posebnom prozoru do koga se dolazi preko ikonice *Overwrite Clock*. U konkretnom slučaju, posle selekcije signala *clk* treba aktivirati ikonicu *Overwrite Clock*. Time se otvori prozor *Clock* u koji se upisuje vreme trajanja periode takta i odnos nule i jedinice u procentima.

Slika 10.12a ilustruje postavljanje signala reset (*rst*) u stanje 1 od 0 – 20 ns, dok je na sl. 10.12b prikazano podešavanje signala takta (*clk*) sa periodom od 100 ns i simetričnim trajanjem niskog i visokog nivoa od po 50%. Talasni oblici pobudnih signala prikazani su na sl. 10.12c.



Slika 10.12. Postavljanje sekvence za pobudne signale *rst* i *clk*

Simulacija se pokreće selektovanjem polja *Simulation* i opcije *Run Functional Simulation*, kao što ilustruje sl. 10.13a. Tokom simulacije otvoren je prozor sa sl. 10.13b u kome se prati tok simulacije. Na kraju, dobija se rezultat u vidu talasnih oblika prikazanih na sl. 10.13c. Podsećamo da se na izlazu dobija 32–bitni signal čija se vrednost proizvoljno menja pri svakoj promeni taktnog signala sa 0 na 1. Stanje svakog bita pojedinačno može da se prati ukoliko se pritisne trougao ispred naziva signala u levom delu glavnog ekrana. Ovo važi za sve signale tipa `standard_logic_vector`.

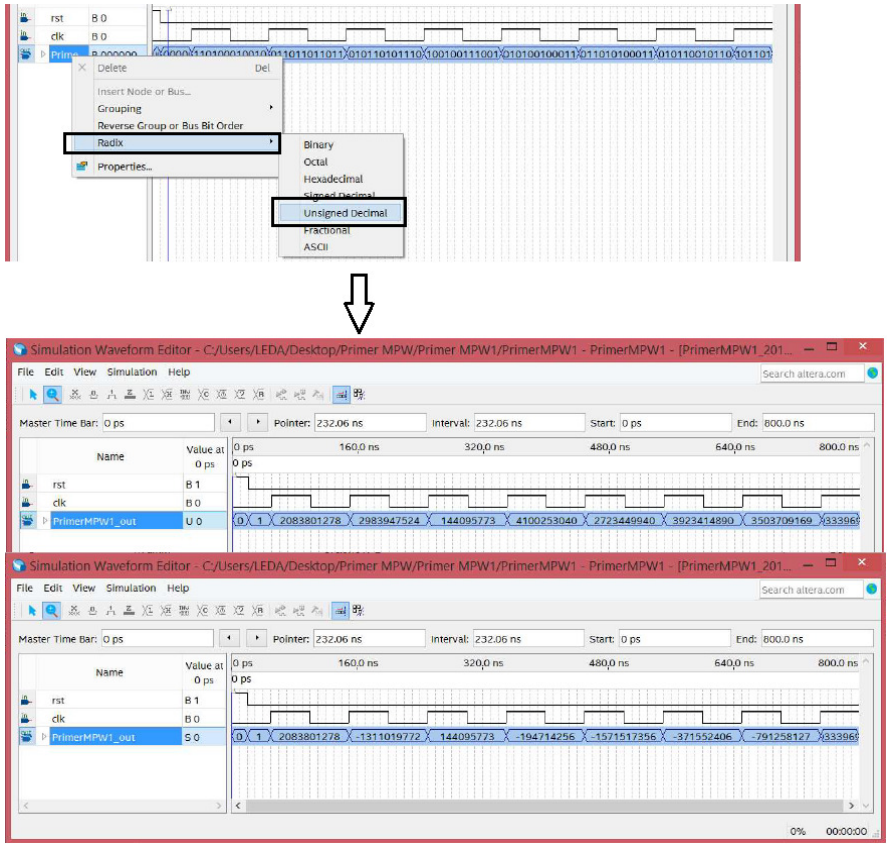


Slika 10.13. Pokretanje (a), praćenje (b) i rezultat (c) funkcionalne simulacije

Binarni prikaz signala obično je nepregledan, tako da je korisno da se prikaže i sa drugačijom brojnom osnovom. Promena brojne osnove veoma je jednostavna. Pri selektovanom signalu, pritisne se desni taster miša, čime se otvara dodatni prozor za editovanje selektovanog signala kao što prikazuje sl. 10.14.

Izborom opcije *Radix*, otvara se padajući meni u kome izaberemo željeni oblik. Ukoliko se opredelimo za decimalni bez znaka (*Unsigned Decimal*) rezultat se automatski prevodi u oblik prikazan u središnjem delu sl. 10.14. Alternativno, da je izabrana opcija decimalni sa znakom (*Signed Decimal*), dobio bi se rezultat prikazan na dnu sl. 10.14.

Ukoliko smo zadovoljni rezultatima simulacije može se preći na programiranje hardvera. Međutim, pre toga, korisno je da se prikažu i drugi oblici rezultata dobijenih sintezom.

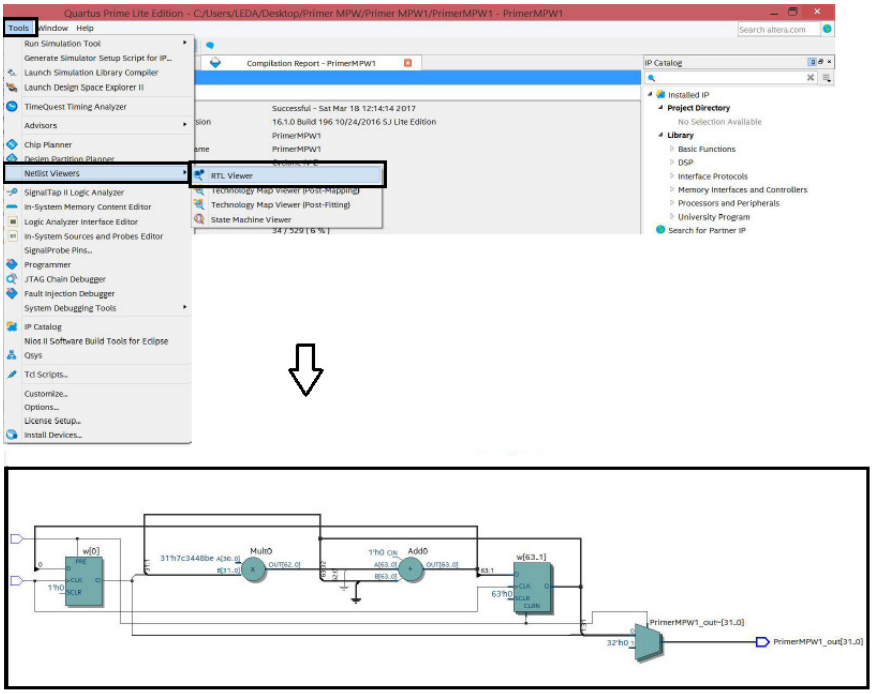


Slika 10.14. Promena brojne osnove prikaza rezultata simulacije

10.1.5. Prikaz rezultata sinteze

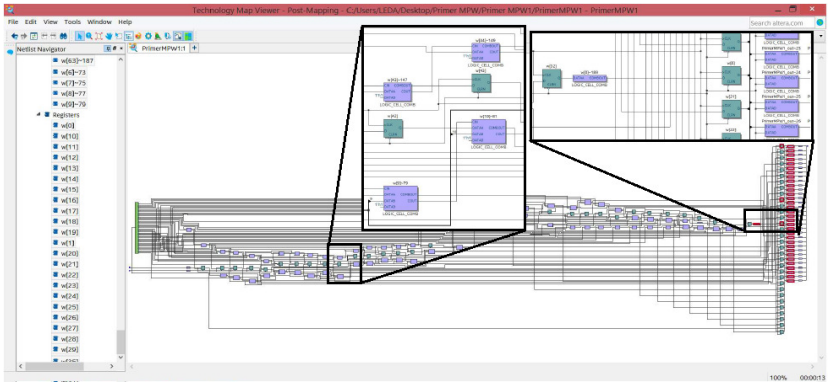
Različiti prikaz rezultata sinteze dostupan je iz padajućeg menija koji se otvara selektovanjem opcije *Tools* u glavnom radnom prozoru (sa sl. 10.1.). Izborom *Netlist Viewers*, nude se prikazi RTL sinteze (*RTL Viewer*), sinteze na tehnološkom nivou (*Technology Map Viewer*) pre (*Post–Mapping*) i posle optimizacije (*Post–Fitting*), kao i sinteze na bazi konačnih automata (*State Machine Vewer*). Svaki od prikaza otvara se u posebnom prozoru, nezavisnom od glavnog radnog prozora programa. Zato svaki sadrži alate za kontrolu prikaza: zumiranje, pometanje slike i sl.

Najpre pogledajmo rezultat RTL sinteze na sl.10.15. Podsećamo da je VHDL kôd sa sl. 10. 6 automatski preveden u logičku šemu na RTL nivou, tako da se prepoznaju blokovi množač, sabirač, multipleksor i dva flipflopa. RTL prikaz je tehnološki nezavisan. Da bi se projekat implementirao na FPGA, potrebno je da se sve logičke strukture prevedu (mapiraju) u

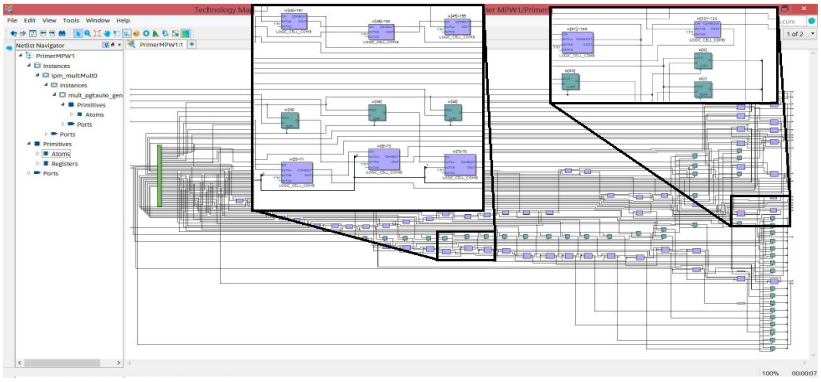


Slika 10.15. Prikaz rezultata RTL sinteze

tehnološke strukture koje realno postoje na FPGA. To su kombinacione ćelije koje, u konkretnom slučaju, nose nazive LOGIC_CELL_COMB ili sekvencijalne ćelije (D flipflop), kao i periferne ćelije IO_OBUF. Nakon alociranja svih potrebnih ćelija, automatski se radi razmeštaj i povezivanje sa ciljem da se optimizuje međusobni položaj potrebnih ćelija kako bi se minimizirao broj zauzetih centralnih blokova i dužine veza. Rezultati direktnog mapiranja prikazani su na sl. 10.16a, a nakon minimiziranja površine na sl. 10.16b U uokvirenim pravougaonicima prikazani su uveličani delovi pojedinih segmenata slike, do kojih se dolazi izborom opcije zumiranja. Ona se aktivira preko ikonice koja prikazuje lupu, u gornjem delu prozora.



a)



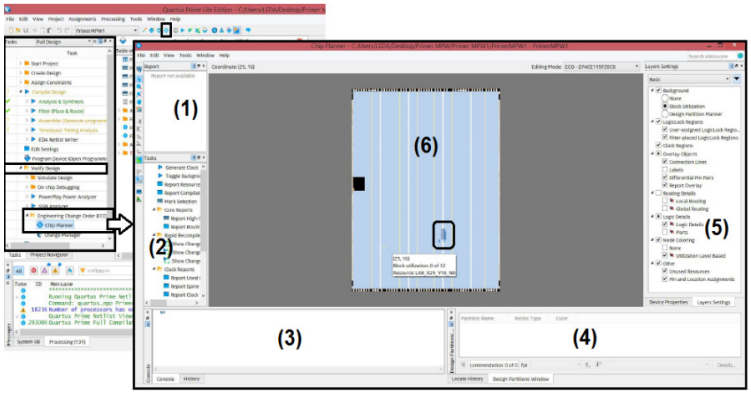
b)

Slika 10.16 Prikaz rezultata mapiranja u tehnološke ćelije pre (a) i posle minimizacije broja zauzetih blokova i veza (b)

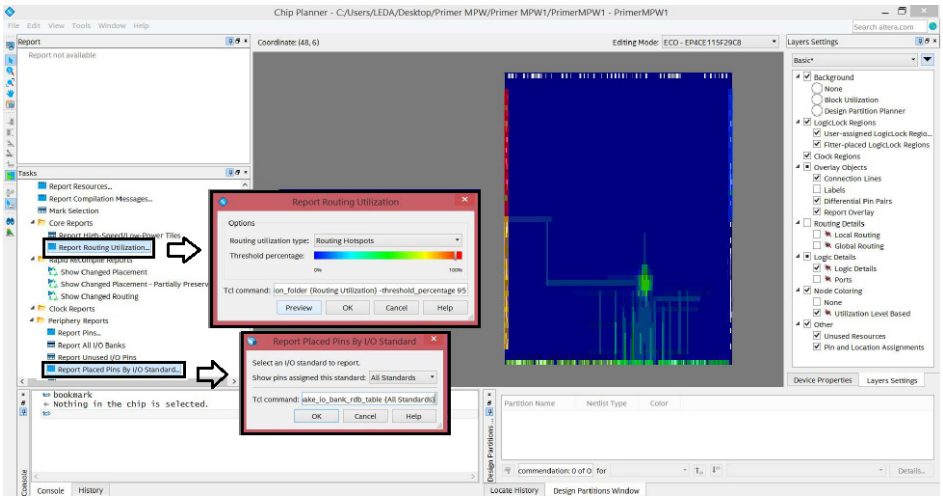
Dobro je videti i fizički raspored blokova na FPGA. Ovo je moguće aktiviranjem prozora *Chip Planner* do koga se najlakše dolazi preko ikonice u polju (2b) sa sl. 10.1 ili selektovanjem *Verify Design>Engineering Change Option>Chip Planner*. Postupak otvaranja ovog prozora i njegov početni izgled prikazani su na sl. 10.17. Jednostavniji način predstavlja selektovanje ikonice u vrhu osnovnog prozora koja je na sl. 10.17 uokvirena. Nešto duži put, sa istim rezultatom, prikazan je preko menija u levom delu sl. 10.17.

Prozor je podeljen u šest oblasti koje su na sl. 10.17 označene brojevima, a po obodu gore i levo su hrizontalni vertikalni meniji. Oblast (1) levo gore, služi za prikaz izveštaja (*Report*), a levo dole (2) izlistani su zadaci (*Task*). U dnu levo (3) je konzola (*Console/History*), a do njega, desno (4) je prikaz manjih celina projekta (*Design Partitions*). U prozoru skroz desno (5) (*Basic*) mogu da se selektuju pojedini nivoi koje želimo da vidimo u centralnom delu prozora (6). U centralnom prozoru prikazuje se fizički izgled FPGA na kome su markirane iskorišćene ćelije. Vidi se da ceo projekat zauzima relativno malu površinu čipa i da je smešten u donjem delu čipa. Ovo je u skladu sa informacijom koja je dobijena posle kompajliranja kroz izveštaj prikazan na sl. 10.8.

Bolji uvid u sve zauzete ćelije na FPGA može da se dobije aktiviranjem *Report Routing Utilatization* u prozoru *Task*, kao što ilustruje sl. 10.18. Tada će pojedini delovi čipa biti obojeni u zavisnosti od stepena zauzetosti u spektru boja od plave (slobodan – 0 %) do crvene (popunjen 100%). Osim toga, u istom obliku, moguće je dobiti i informaciju vezanu za stepen zauzetosti pinova. Do nje se dolazi ukoliko se, iz *Task* prozora, sukcesivno, aktivira polje *Report Placed Pins by I/O Standard.* Tada se otvara mali



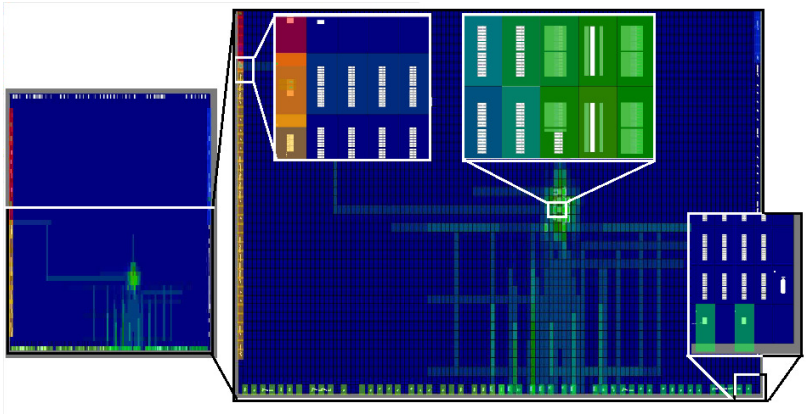
Slika 10.17. Postupak otvaranja i početni izgled prozora *Chip Planner*



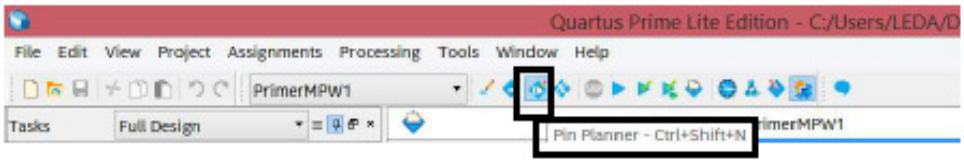
Slika 10.18. Postavljanje parametara za bolju preglednost zauzetosti resursa na čipu

prozor koji omogućava da se selektuju pinovi određenog tipa. U primeru sa sl. 10.18 izabrani su svi standardni pinovi (*All Standards*). Da ne bi bilo zabune, ponavljamo, da se izgled dobijen u pozadini sl. 10.18 dobija nakon aktiviranja oba prozora: *Report Routing Utilatization* i *Report Placed Pins by I/O Standard*.

U centralnom delu prozora *Chip Planner*, prikazaće se izgled čipa u kome su pojedina centralna i periferna polja obojena u skladu sa stepenom



Slika 10.19. Ilustracija opcija uvećavanja (*zoom*) u prozoru *Chip Planner*



Slika 10.20. Aktiviranje programa *Pin Planner*

njihovog zauzeća. Detalji mogu da se uvećaju selekcijom ikonice koja prikazuje lupu, iz levog vertikalnog menija u prozoru *Chip Planner*. Rezultat ove aktivnosti prikazan je na sl. 10.19. Ceo čip prikazan je u krajnjem levom delu slike, a donja polovina (odvojena belom linijom) prikazana je uvećana u centralnom delu slike. I ova slika može dodatno da se uveća na isti način (aktiviranjem ikonice sa lupom). Delimičan prikaz uvećanih detalja iz: levog gornjeg perifernog dela, centralnog dela i desnog donjeg perifernog dela prikazan je na istoj slici. Radi lakšeg raspoznavanja, svi uvećani delovi, naknadno su uokvireni.

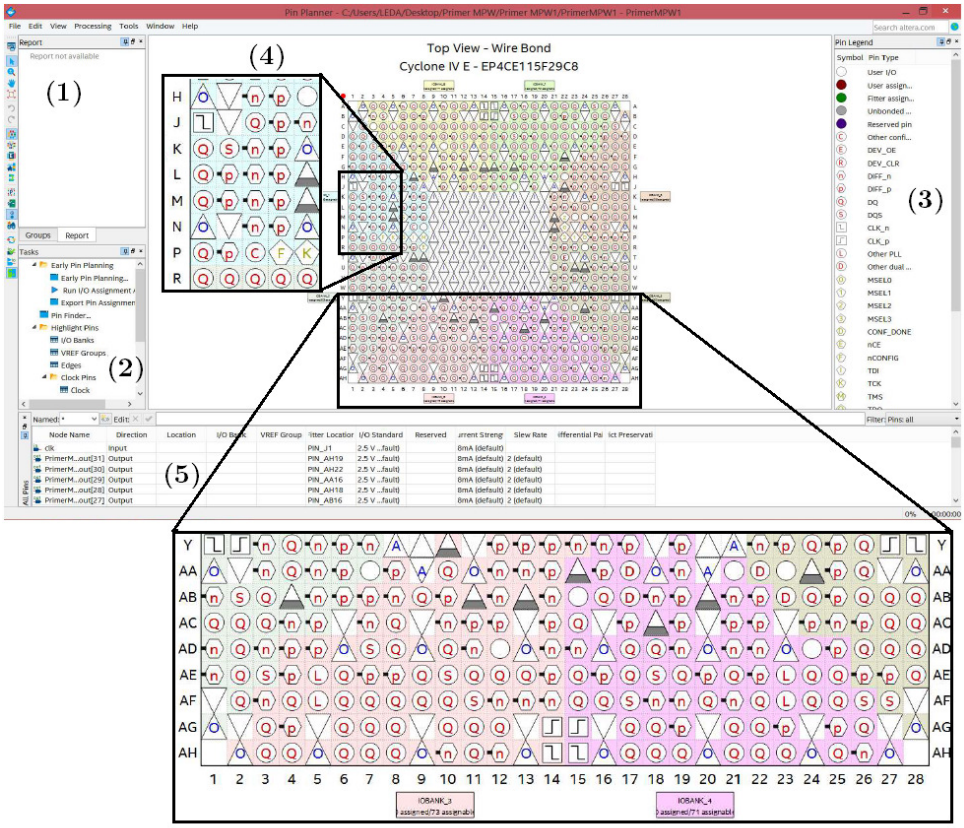
Pozicija iskorištenih ćelija na FPGA zavisi od pozicije upotrebljenih pinova. U ovom projektu nismo definisali raspored pinova. Umesto toga, prihvaćen je podrazumevani raspored, koji je ponudio alat za projektovanje. Naravno, veoma je korisno da znamo upotrebljeni raspored pinova.

Usvojeni raspored pinova može da se pregleda i, eventualno modifikuje, u programu za editovanje pinova koji se zove *Pin Planer*. Iako se do njega može doći preko *Task* potprozora u glavnom *Quartus Prime* prozoru, mnogo je jednostavnije da se on aktivira selektovanjem ikonice iz horizontalnog menija u vrhu ekrana osnovnog prozora *Quartus Prime*, kao što pokazuje sl. 10.20.

Tada se otvara novi prozor na čijem vrhu piše *Pin Planner*. Izgled ovog prozora prikazan je na sl. 10.21. Prozor je podeljen na pet sekcija. Levo su dve, označene sa (1) i (2), u kojima se prikazuju izveštaji (*Report*) i zadaci (*Task*), redom. Desno (3), prikazana je legenda koja objašnjava simbole pojedinih vrsta pinova. U centralnom delu (4), prikazan je izgled svih pinova. Oni su raspoređeni u obliku matrice čije su kolone označene brojevima od 1 do 28, a redovi označeni slovima od A do AH(posle slova Y, oznake se nastavljaju sa AA do AH). U dnu je dat spisak signala sa nazivima specificiranim u projektu i oznakom lokacije pinova. Konkretno, na sl. 10.21 prikazani su:

- signal takta sa nazivom *clk* povezan je za pin na poziciji J1;

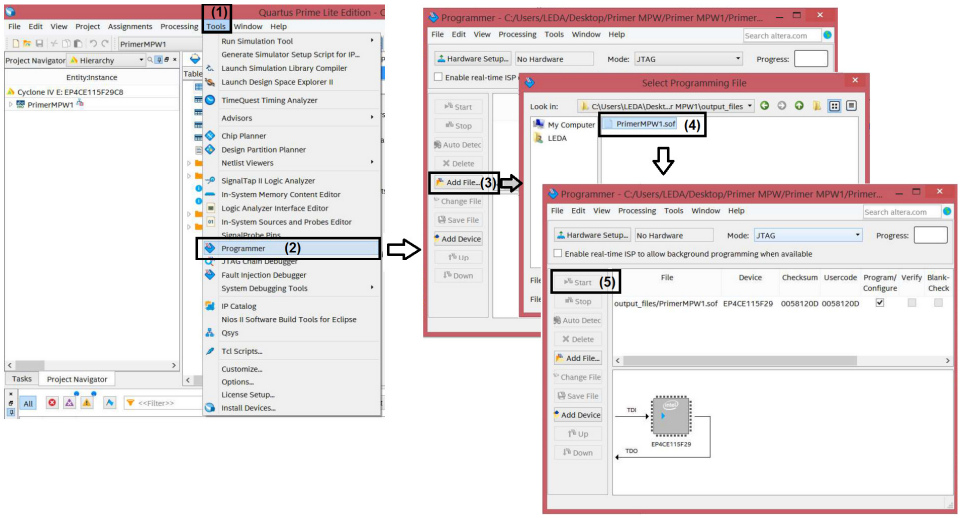
- 31. bit izlaznog signala: *PrimerMPW2_Out(31)*, vezan je za pin AH19;
- 30. bit izlaznog signala: *PrimerMPW2_Out(30)*, vezan je za pin AH22 (naznačeno linijom koja ukazuje na položaj pina u zumiranom delu slike);
- 29. bit izlaznog signala: *PrimerMPW2_Out(29)*, vezan je za pin AH16;
- 27. bit izlaznog signala: *PrimerMPW2_Out(28)*, vezan je za pin AB16 (naznačeno linijom do zumiranog dela slike).



Slika 10.21. Pokretanje *Pin Planner* alata i prikaz rasporeda pinova (uokvireni su delovi na koje je primenjena opcija uvećanja)

Sada je jasno da je ceo projekat pozicioniran u donji deo ekrana zato što je 32–bitni izlazni signal alociran za pinove koji se nalaze u redovima od Y do AH. Linija koja se na sl. 10.18 i sl. 10.19 vidi kao veza koja dolazi do pina u sredini levo, povezuje MPW RG sa signalom takta koji se nalazi na poziciji J1.

Poslednji korak je programiranje FPGA. U tu svrhu koristi se alat *Progremmer*. On se pokreće iz osnovnog prozora, ispod padajućeg menija *Tools*, kao što ilustruju prva dva koraka na sl. 10.22. U prozoru *Programmer* treba selektovati opciju *Add File* u meniju sa leve strane (3). U novootvorenom prozoru za pretraživanje, sa nazivom *Select Programming File*, prikažu se svi raspoloživi fajlovi sa nastavkom *.sof*. Ovi fajlovi se generišu posle kompajliranja i smeštaju u direktorijum *output files* koji se automatski otvara u radnom direktorijumu. U konkretnom slučaju, prikazanom kao korak 4 na sl. 10.22, postoji samo jedan fajl sa nazivom *PrimerMPW1.sof*. On sadrži sve potrebne informacije o projektu, uključujući i naziv hardvera na koji se prenosi projekat. Fizički, FPGA se programira aktiviranjem dugmeta *Start*, kao što ilustruje korak 5 na sl. 10.22.



Slika 10.22. Pokretanje alata za programiranje FPGA

10.2. SEMAFOR ZA TENIS

Nakon detaljnog opisa korišćenja *Quartus Prime* alata za projektovanje FPGA, demonstriraćemo njegovu primenu na nešto složenijem primeru. Razmotrićemo realizaciju integrisanog kola namenjenog za automatski prikaz rezultata teniskog meča na semaforu [Ili14, Ili15].

10.2.1. Opis projektnog zadatka

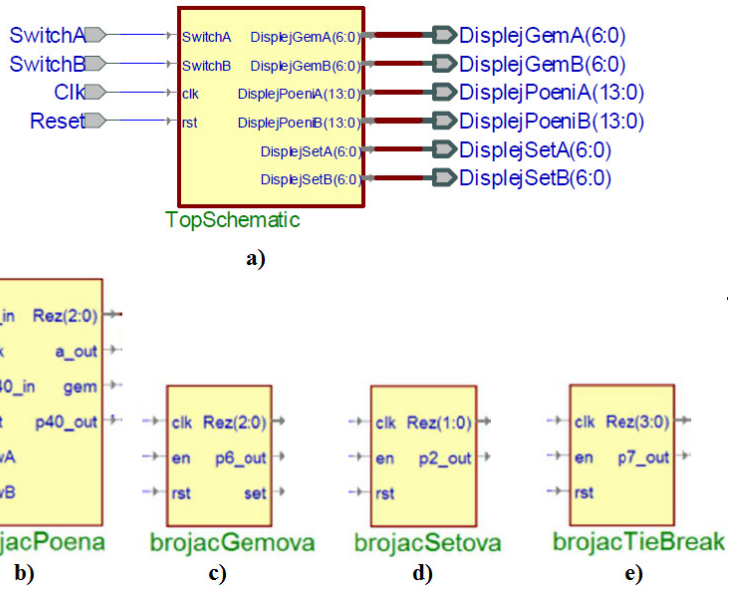
Cilj je da se projektuje sistem u kome sudija, pritiskom na jedan od dva tastera, dodeljuje poene jednom ili drugom igraču, a da se rezultat automatski generiše i prikazuje na semaforu sa osam cifara: po dve za stanje poena, i po jedan za prikaz stanja gemova i setova za svakog igrača. Alternativa je da sudija umesto tastera koristi palicu koja se pomera levo ili desno prema igraču koji osvoji poen. Nezavisno od načina realizacije sudijske konzole, sa stanovišta signala, to znači da se generiše impuls kada jedan od igrača dobije poen. Akumulator poena sabira osvojene poene, generiše signale za prikaz trenutnog stanja poena na dvocifrenom sedmosegmentnom displeju za svakog igrača. Stanje poena prikazuje se ciframa 00,15, 30 40 i slovom A. Osim toga, potrebno je da se upoređuju stanja poena oba igrača sa prethodnim i generiše signal kada se osvoji gem. Ovaj signal pobuđuje akumulator gemova u kome se inkrementira vrednost, generišu signali za prikaz trenutnog stanja gemova na displeju, porede vrednosti i generiše signal posle ispunjenog uslova za dobijanje seta. Na sličan način, brojač setova inkrementira stanje po osvojenom setu, generiše signale za pobudu displeja za svakog igrača, poredi stanje i generiše signal koji označava završetak igre. Osim toga, predviđeno je da sudija kontroliše i taster *Reset* preko koga se anulira rezultat.

10.2.2. Realizacija projekta

Realizaciji projekta pristupa se uz poštovanje osnovnih principa projektovanja: hijerarhijska podela, regularnost, modularnost i lokalizovanje (Videti poglavlje 1.3). Imajući u vidu da su kod PLD modularnost i lokalizovanje već ugrađeni, potrebno je podeliti zadatak na manje celine sa ciljem da se, naročito na nižim nivoima, koristi što više istih blokova.

Prateći osnovnu logiku projektnog zadatka, on je podeljen na sledeće osnovne celine: *brojač poena*, *brojač gemova*, *brojač setova*, *brojač poena u tie break–u*, *kontrola prikaza rezultata*. Na nižem hijerarhijskom nivou definisani su pomoćni blokovi: *komparator veće*, *komparator jednako*, *oduzimač*, *generator jednog impulsa* (služi da emulira taster za dodelu poena), *kontrola prikaza broja poena*, *kontrola prikaza broja gemova* i *kontrola prikaza broja setova*. Naravno, iznad svih je *vrhovna kontrolna logika* (*Top level*) koja upavlja radom celog sistema. Na sl. 10.23.a prikazan je entitet kontrolne logike, nazvane *TopSchematic*. Ulazni signali koji simuliraju tastere nazvani su *SwitchA* i *SwitchB*, a osim njih, prikazani su signal za resetovanje (*Reset*) i taktni signal nazvan *Clk*. (Taktni signal olakšava testiranje, odnosno verifikaciju.) Izlazne signale predstavljaju sedmobitni signali za kontrolu sedmosegmentnih displeja i to *DisplejSetA*, *DisplejSetB*, *DisplejGemA* i *DisplejGemB*, kao i 14–bitni signali za kontrolu dvocifrenih displeja broja poena nazvani *DisplejPoeniA* i *DisplejPoeniB*.

Prilikom označavanja, slova A i B koriste se kako bi ukazali da se odnose na igrača A, odnosno igrača B. U okviru *TopSchematic* bloka, između ostalog, odlučuje se o ulasku u *tie break*.



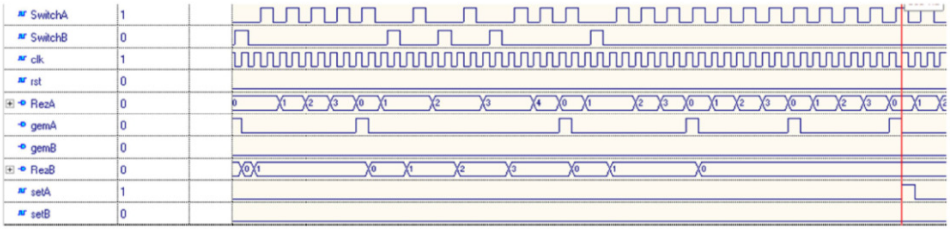
Slika 10.23. Osnovni entiteti celog projekta: a) kontrolna logika (*TopSchematics*), b) brojač poena, c)brojač gemova, d) brojač setova i e) brojač u *tie break*–u

Entiteti osnovnih blokova na nižem hijerarhijskom nivou: *brojač poena*, *brojač gemova*, *brojač setova* i *brojač poena u tie break–u* prikazani su na sl. 10.23b, sl. 10.23c, sl. 10.23d i sl. 10.23e, redom. Ponašanje svih navedenih blokova opisano je na funkcionalnom nivou primenom VHDL jezika za opis hardvera. Svaki je nezavisno verifikovan simulacijom.

Brojač poena treba da omogući da se, pri svakom inkrementiranju signala koji dolazi iz tastera, generiše izlazni signal koji će se na semaforu prikazati u obliku: 0, 15, 30, 40 ili A. Ulazni signali *swA*, odnosno *swB* predstavljaju izlaz iz tastera. Dovođenjem logičke jedinice preko signala *rst* brojač se dovodi u početno, nulto stanje. Osim ovih, postoje i ulazni signali koji služe za detektovanje stanja protivničkog igrača. To su signali: *a_in*, koji ukazuje da protivnički igrač ima prednost, i signal *p40_in*, kojim se ukazuje da suparnik ima 40 poena. Brojač poena generiše tri jednobitna i jedan trobitni signal na izlazu. Jednobitni signali su: *a_out*, koji se postavlja na 1 ukoliko igrač ima prednost, *p40_out*, koji se postavlja na 1 kada igrač osvoji 40 poena, i *gem* koji generiše impuls kada igrač osvoji gem. Trobitni izlazni signal *rezultat* predstavlja izlaz iz trobitnog brojača. On se vodi u blok za kontrolu prikaza broja poena preko konvertora stanja brojača u stanja 0, 15, 30, 40 i A na sedmosegmentnom dvocifrenom displeju. Ako je vrednost brojača „40", a vrednost signala *p40_in* nula, dolaskom sledećeg signala, igrač osvaja gem i generiše se impuls na signalu *gem*. Ukoliko je vrednost brojača „A" i vrednost signala *p40_in* jedinica, novi poen donosi igraču osvojeni gem. Ukoliko u tom stanju, protivnik osvoji poen, dolazi se do izjednačenja, odnosno oduzima se u brojaču jedan poen. Suštinski, brojač poena broji do tri, a četvrti poen predstavlja osvojen gem i tada se brojač resetuje na nulu. Kada je vrednost brojača jednaka tri, tada se izlazni signal *p40_out* postavlja na jedinicu.

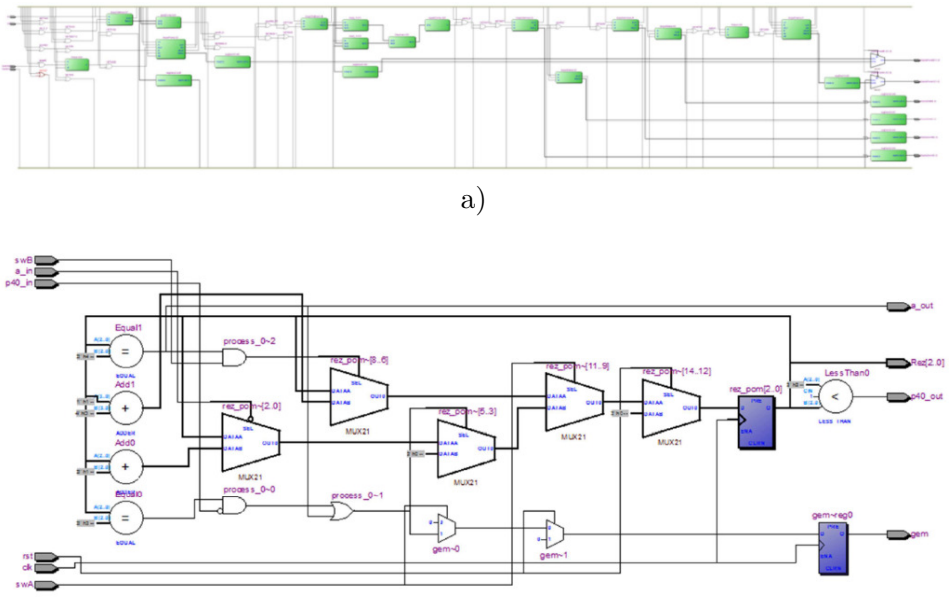
Brojač gemova pobuđen je signalom *gem* preko ulaza *en*. To je trobitni brojač koji broji do šest. Kada se registruje šesti gem generiše se stanje 1 na izlazu *p6_out*. Upoređuje se stanje sa suparnikom i ukoliko nije registrovan *tie brak*, generiše se impuls na izlazu *set*. Istovremeno se resetuje brojač. Trenutno stanje brojača prikazuje trobitni izlaz *Rez* koji se vodi do *kontrole prikaza rezultata*, odnosno *kontrole prikaza broja gemova* u kome se konvertuje u sedmobitni signal za pobuđivanje sedmosegmentnog displeja.

Signal *set* koji dolazi iz *brojača gemova* pobuđuje *brojač setova* preko ulaza *en*. U konkretnom slučaju predviđeno je da broji do dva dobijena seta. Brojač je realizovan kao dvobitni. Na izlazu generiše trenutno stanje brojača dvobitnim signalom Rez i jednobitni indikator da li je neki od igrača dobio dva seta, nazvan *p2_out*.



Slika 10.24. Rezultat verifikacije rada sistema za praćenje teniskog meča

Kada se ostvari uslov za *tie break*, odnosno, ukoliko je rezultat u gemovima izjednačen do šestog gema, aktivira se blok *brojač poena u tie break–u*. Radi se o trobitnom brojaču. Gem dobija igrač koji prvi osvoji sedam poena sa razlikom većom od jednog gema. Nakon toga, gem osvaja igrač koji prvi napravi skor za dva veći od protivnika. Izlaz iz tastera dovodi se na *en* ulaz ovog brojača. Na izlazu se stanje brojača prikazuje preko trobitnog signala *Rez*. Osim toga, ovaj blok generiše stanje 1 kada igrač osvoji 7 poena na signalu nazvanom *p7_out*.



a)

b)

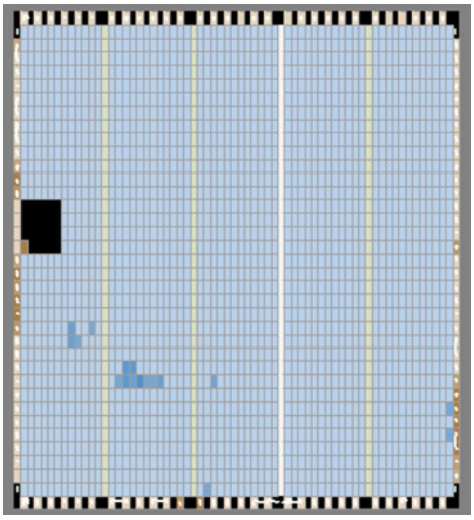
Slika 10.25. RTL prikaz: a) celog projekta i b) bloka brojač poena

Naravno, osim osnovnih entiteta sa sl.10.23 opisani su u VHDL–u i ostali pomoćni entiteti kao što su blokovi za: kontrolu prikaza rezultata, upoređivanje, oduzimanje i sl. Opis svakog od njih verifikovan je simulacijom, a sl. 10.24 prikazuje deo rezultata simulacije celog sistema. Prikazan je detalj u kome igrač A osvaja gem i set.

Verifikovani VHDL kodovi uneti su u *Quartus* na način opisan u poglavlju 10.1.2. Zatim je, u prozoru *Project Navigator* (potprozor 3 na sl. 10.1), selektovan *TopSchematic.vhd* kao entitet na najvišem hijerarhijskom nivou i pokrenuto kompajliranje, na način opisan u poglavlju 10.1.3.

Po uspešno završenom kompajliranju dostupan je izveštaj o zauzetim ćelijama. Pored toga, kao što je rečeno u poglavlju 10.1.5, moguće je generisati različite oblike prikaza rezultata. Na sl. 10.25a prikazan je izgled projekta na RTL nivou, dobijen pokretanjem *RTL Viewer* programa. Slika 10.25b daje RTL prikaz bloka *brojač poena*. Naravno, mogu se videti i ostale vrste prikaza koje nudi *Netlist Viewer* kao što su *Technology Map Viewer*, *Post–Mapping* i *Post–Fitting* prikazi.

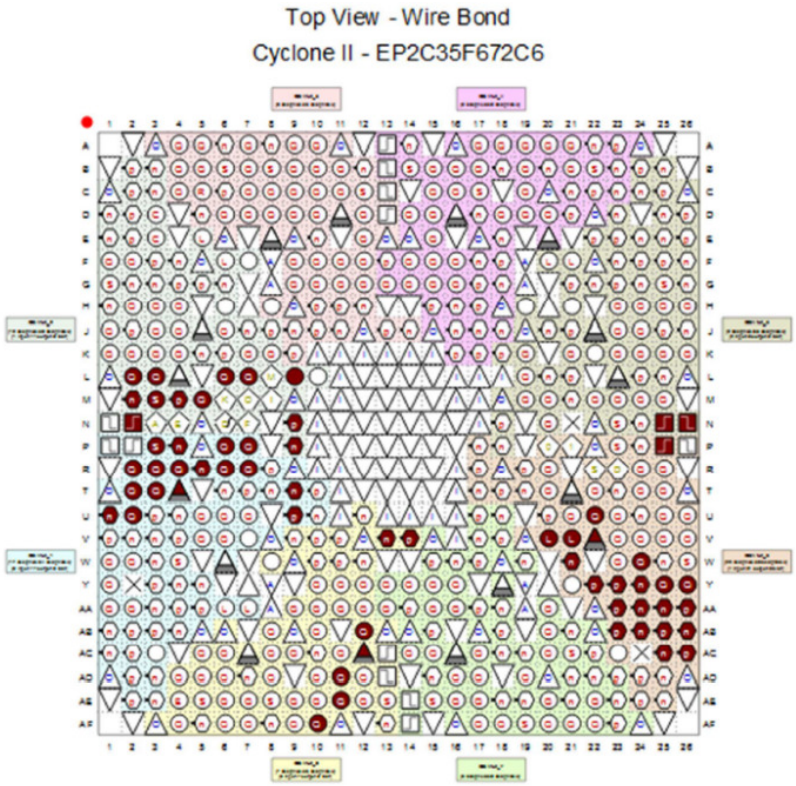
Uvek je zanimljivo videti i fizički razmeštaj na čipu do koga se dolazi aktiviranjem programa *Chip Planner*. Ceo projekat je realizovan na *DE2* razvojnoj ploči na kojoj se nalazi FPGA sa oznakom EP2C3F672C6.



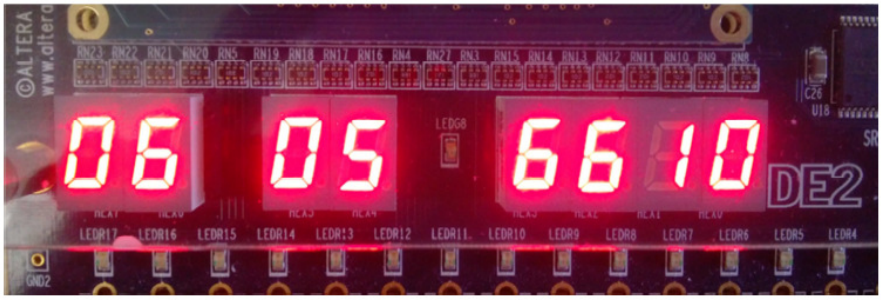
Slika 10.26. Raspored zauzetih resursa na čipu EP2C3F672C6

Upotrebljeno je ukupno 170 od 33216 logičkih elemenata i 32 od 33216 registara, tako da je zauzeto manje od 1% svih resursa. Iako su ovo podaci koji se dobijaju u izveštaju nakon kompajliranja, njihov pravi značaj postaje vidljiv tek pozivom *Chip Planner* programa. Na sl. 10.26 prikazan je fizički izgled zauzetih ćelija na čipu.

Na osnovu analize entiteta na najvišem nivou, sl. 10.23a, očigledno je da projekat zahteva upotrebu 60 pinova. Od toga su četiri ulazna dok je 56 izlaznih pinova koje treba povezati sa displejem. Njihov raspored postaje vidljiv aktiviranjem programa *Pin Planner* na način ilustrovan na sl. 10.20. Kao što je ranije rečeno, raspored može da se koriguje, a i u ovom slučaju, prihvaćeno je rešenje koje je uslovljeno povezivanjem sa ugrađenim displejem i raspoloživim tasterima na razvojnoj ploči *DE2* [Alt06] na kojoj će se demostrirati projekat. Konačni raspored pinova prikazan je na sl. 10.27.



Slika 10.27. Raspored pinova



Slika 10.28. Demonstracija rada semafora za tenis na *DE2* ploči

Poslednja faza realizacije projekta je programiranje FPGA. Postupak je objašnjen u poglavlju 10.1.5 i ilustrovan na sl. 10.22, tako da nema potrebe za dodatnim objašnjenjima. Umesto toga, demonstriraćemo rad sistema na razvojnoj ploči *DE2*. Naime, osim FPGA (u našem slučaju EP2C3F672C6), ploča sadrži i potrebne tastere kao i sedmosegmentne displeje, povezane sa određenim pinovima na čipu. Pored toga, na ploči je ugrađen i potreban hardver za povezivanje sa računarom. (Za detaljni opis svih mogućnosti koje ova ploča nudi, upućujemo čitaoce na korisničko uputstvo [Alt06]). Isprogramirani FPGA ponašao se u skladu sa rezultatima dobijenim simulacijama.

Na sl. 10.28 prikazan je deo ploče na kome se nalazi displej. Ilustrovana je situacija u kojoj se igra *tie break* u drugom setu. Prve dve cifre (06) prikazuju vrednost poena za prvog igrača, druge dve cifre vrednosti poena (05) za drugog igrača. Peta i šesta cifra prikazuju vrednost gemova prvog i drugog igrača. S obzirom da su obe cifre 6, to znači da se igra *tie break*. Poslednje dve cifre, sedma i osma, prikazuju vrednost osvojenih setova prvog (1) i drugog igrača (0).

Literatura

[Alt06] *Altera Corporation*, *DE2* Development and Education Board – User Manual, [On–line] ftp://ftp.altera.com/up/pub/Webdocs/DE2_User Manual.pdf, (accessed April 2017)

[Gor03] Goresky M., Klapper A., „Efficient Multiply–with–Carry Random Number Generators with Maximal Period", *ACM Transactions on Modeling and Computer Simulation,* Vol. 13, No. 4, October 2003, pp. 1–12.

[Ili14] Ilijin S., Petković P., „Kontrolna logika za praćenje i prikaz rezultata teniskog meča", *Zbornik radova 58. Konferencije za elektroniku,*

telekomunikacije, računarstvo, automatiku i nuklearnu tehniku, ETRAN 2014, Vrnjačka Banja 2.–5. Jun 2014, ISBN 978–86–80509–70–9, pp. EL1.3.1–6.

[Ili15] Ilijin S., Petković P., „Implementation of Control Logic in the Scoreboard of Tennis", *Serbian Journal Of Electrical Engineering*, Vol. 12, No.2, June 2015, pp. 219–236, DOI: 10.2298/SJEE1502219I.

[Mar03] Marsaglia, G., "Random Number Generators", *Journal of Modern Applied Statistical Methods*, (2003), 2(1), pp. 2–13.

[Mis08] Mišić G., „Alati za projektovanje", *Seminarski rad*, Elektronski fakultet Univerzitet u Nišu, 2008.

[Tay16a] A. Taylor (2016, July 14), „10 FPGA Design Techniques You Should Know" *EETimes*, [On–line] Available: http://www.eetimes.com/document.asp?doc_id=1330128 [2016, July, 20]

[Tay16b] A. Taylor (2016, June 10), „10 Ways To Program Your FPGA", *EETimes*, [On–line] Available: http://www.eetimes.com/document.asp?doc_id=1329857 [2016, July, 15]