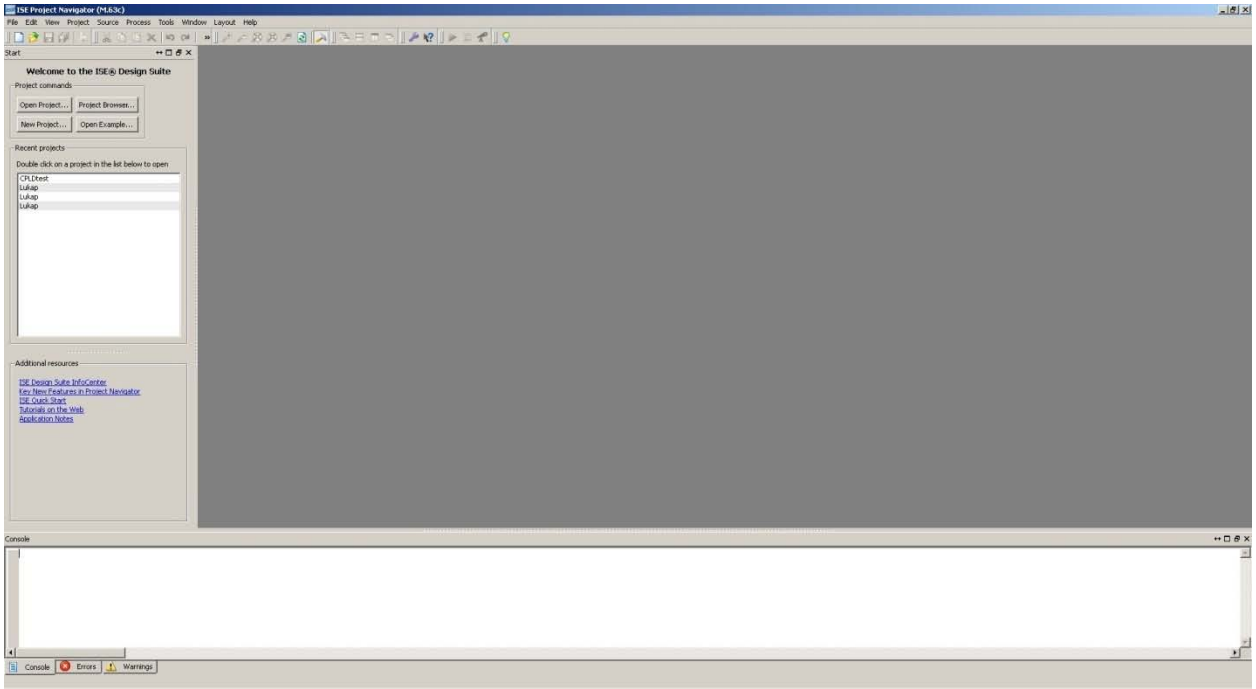
PROGRAMIRANJE KOMUNIKACIONOG HARDVERA
– Poglavlje 3 (deo 2) –

3.3 ISE softverski paket

ISE softverski paket predstavlja razvojno okruženje za FPGA čipove kompanije Xilinx. Trenutno poslednja verzija je 14.4, koja je izdata u decembru 2012. U okviru ovih skripti biće opisana verzija 12.2 softverskog paketa ISE. Opis razvojnog okruženja će pratiti kreiranje jednog konkretnog projekta, formiranje sastavnih delova projekta i na kraju kompajliranje kreiranog projekta. Na ovaj način će na praktičnom primeru biti prikazani osnovni delovi ISE paketa i načini njihovog korišćenja da bi se kreirala jedna hardverska implementacija koja treba da se spusti na FPGA čip. Napomenimo još da definicije pojedinih pojmova i tehnika neće biti ponovo date u ovoj sekciji u slučaju da su prethodno objašnjeni u sekciji 3.2, radi izbegavanja ponavljanja.

3.3.1. Glavni prozor ISE aplikacije

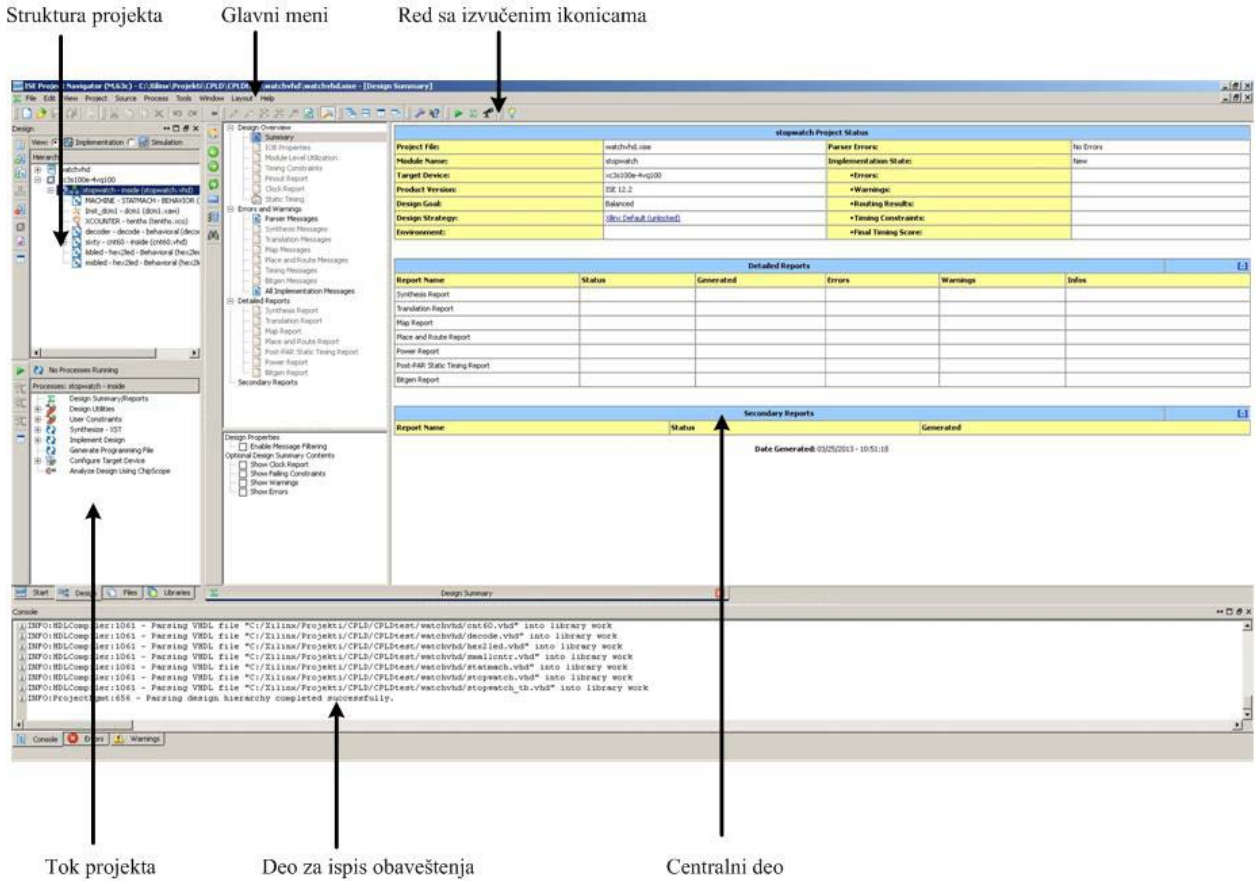
Po pokretanju ISE softvera, dobija se sledeći prikaz prozora u kome je otvorena ISE aplikacija (slika 3.3.1.1.). Glavni prozor u slučaju kada nije otvoren nijedan projekat nudi sa leve strane meni za kreiranje novog projekta (**New Project**), kao i otvaranje postojećeg projekta (**Open Project**). **Project Browser** opcija služi za kreiranje, modifikovanje i brauzovanje liste projekata tako da se može brzo prebacivati (brauzovati) sa projekta na projekat. **Open Example** opcija daje mogućnost otvaranja primera projekta koji su sastavni deo ISE paketa i pre svega služe kao svojevrsni tutorijal za nove korisnike. Sve četiri navedene opcije se nalaze i pod **File** opcijom glavnog menija. Ispod navedene četiri opcije, takođe sa leve strane glavnog prozora, se nalazi lista poslednje otvaranih projekata u ISE paketu. Dvostrukim klikom na neki od projekata će biti otvoren dotični projekat.



Slika 3.3.1.1. – Glavni prozor ISE aplikacije kada nije otvoren nijedan projekat

Otvorimo, upotrebom opcije **Open Example**, projekat pod nazivom **watchvhd** radi prikaza glavnog prozora kada je otvoren projekat radi objašnjenja pojedinih delova glavnog

prozora. Izgled glavnog prozora kada je otvoren projekat je prikazan na slici 3.3.1.2. Glavni prozor sadrži glavni meni, red sa izvučenim ikonicama, struktura projekta, tok projekta, centralni deo i deo za ispis obaveštenja. Kao što vidimo po sastavnim delovima, izgled prozora je strukturno identičan izgledu prozora Quartus softverskog paketa.



Slika 3.3.1.2. – Glavni prozor ISE aplikacije kada je otvoren projekat

Glavni meni grupiše sve opcije u srodne grupe i preko njega se može pristupiti svim opcijama koje nudi ISE. Grupe opcija su raspoređene u **File**, **Edit**, **View**, **Project**, **Source**, **Process**, **Tools**, **Window**, **Layout** i **Help**. **File** grupa opcija sadrži osnovne operacije za kreiranje/otvaranje/snimanje fajlova i projekata. **Edit** grupa opcija se odnosi na opcije koje se koriste u editovanju otvorenih fajlova poput kopiranja, *paste* opcije, *undo* opcije i dr. **View** grupa opcija daje mogućnost podešavanja prozora, na primer prikaz na celom ekranu, zumiranje, prikaz rednih brojeva linija koda u otvorenom VHDL fajlu i dr. **Project** grupa opcija se koristi za menadžment projekta, gde se dodaju novi fajlovi u projekat, podešavaju parametri projekta i dr. **Source** grupa opcija omogućava kontrolu nad izvorišnim (*source*) fajlovima poput postavljanja otvorenog fajla kao top level entiteta, uvida u svojstva izvorišnog fajla (*properties*). **Process** grupa opcija nudi mogućnost pokretanja i podešavanja procesa kompajliranja. **Tools** grupa opcija omogućava aktivaciju posebnih alata poput programera koji programira FGPA čip, alati za uvid u vizuelni prikaz implementiranog dizajna na FPGA čipu, alati za analizu potrošnje snage dizajna i dr. **Window** grupa opcija služi za menadžment prozora. **Help** grupa opcija, kao što i sam naziv kaže, se koristi za nalaženje pomoći u vidu dodatne dokumentacije.

Red sa izvučenim ikonicama sadrži najvažnije opcije iz glavnog menija u vidu ikonica čime se omogućava brz pristup najvažnijim opcijama. Neke od najvažnijih opcija koje su stavljene u ovaj red po difoltu su otvaranje i snimanje fajlova, pokretanje potpunog kompajliranja, prikaz sažetog izveštaja dizajna i dr. Pokretanjem pokazivača iznad neke od ikonica u oblačiću se ispisuje naziv dotične ikonice, odnosno opcije koju dotična ikonica predstavlja.

Struktura projekta je deo glavnog prozora u kome se nalaze informacije o strukturi projekta. Na vrhu ovog dela postoji izbor za prikaz projekta - **Implementation** i **Simulation**. **Implementation** prikaz daje hijerarhijski prikaz projekta gde se u obzir uzimaju samo implementacioni fajlovi koji se koriste u procesu kompajliranja za generisanje konfiguracionog fajla kojim će se programirati FPGA čip. **Simulation** prikaz daje hijerarhijski prikaz projekta gde se u obzir uzimaju samo fajlovi potrebni za obavljanje procesa funkcionalne ili vremenske simulacije. U slučaju **Simulation** prikaza pojavljuje se padajući meni sa opcijama tipa simulacije. **Behavioral** opcija se koristi za funkcionalnu simulaciju, a preostale tri (**Post-Translate**, **Post-Map** i **Post-Route**) za vremensku simulaciju. U oba slučaja se dobija hijerarhijski prikaz odgovarajućih delova projekta (fajlova) i njihovih međusobnih relacija. Sa leve strane su izvučene ikonice za najčešće korišćene operacije u manipulisanju strukturom projekta, poput ikonice za postavljanje selektovanog fajla kao top level entiteta ili ikonice za dodavanje novog fajla u projekat.

Tok projekta predstavlja deo glavnog prozora u kome se prati tok kompajliranja. U slučaju da je u strukturi projekta selekovan **Implementation** prikaz i selektovan top level entitet, tok projekta daje prikaz etapa procesa kompajliranja. Opcije koje imaju dve strelice povezane u krug predstavljaju korake kompajliranja koji se mogu pokrenuti dvostrukim klikom. Svaka od faza ima dodatne opcije koja su vidljiva raširivanjem prikaza faze. U slučaju da je selektovan neki od entiteta koji nisu top level entitet, onda tok projekta daje samo opcije moguće za selektovani fajl poput provere sintakse u selektovanom fajlu, a koraci procesa kompajliranja nisu vidljivi jer dotični selektovani fajl nije top level entitet. U slučaju **Simulation** prikaza, tok projekta daje mogućnost za pokretanje funkcionalne ili vremenske simulacije (zavisno od toga koja je opcija izabrana u padajućem meniju) za selektovani fajl koristeći integrisani **Isim** simulator.

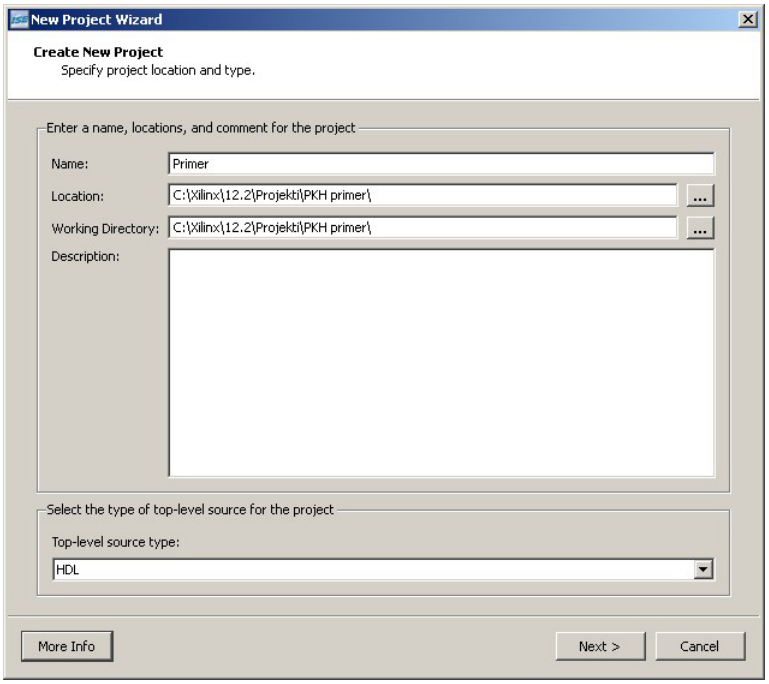
Ispod toka projekta postoji meni sa četiri taba (**Start**, **Design**, **Files**, **Libraries**). Tab **Start** umesto strukture i toka projekta daje prikaz osnovne četiri opcije i liste nedavno otvaranih projekata, odnosno meni sa leve strane koji je prikazan na slici 3.3.1.1. Tab **Design** daje prikaz strukture i toka projekta kao što je prikazano na slici 3.3.1.2. Tab **Files** daje listu svih fajlova uključenih u projekat. Tab **Libraries** daje listu svih korisničkih biblioteka uključenih u projekat. Po difoltu je uvek prisutna **work** biblioteka.

Centralni deo prozora se koristi za otvaranje tekućih fajlova u kome se može vršiti editovanje fajlova, čitanje izveštaja, i sl. Na primer, u ovom delu se otvaraju VHDL fajlovi, tj. preciznije otvara se editor koji omogućava editovanje VHDL fajlova.

Deo za ispis obaveštenja se koristi za ispis raznih obaveštenja, prevashodno onih koji se odnose na tok kompajliranja. Tu se prikazuju obaveštenja o greškama, o toku aktiviranog kompajliranja, upozorenja i sl.

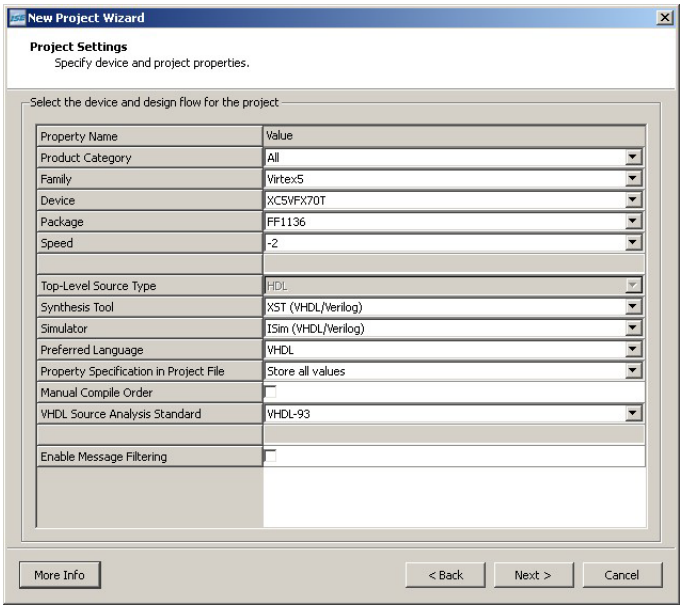
3.3.2. Kreiranje projekta

Izaberimo opciju **File->New Project** iz glavnog menija. Pokretanjem ove opcije otvaramo dijalog za kreiranje novog projekta. Napomena: na slikama su prikazani podaci koje treba unositi prilikom kreiranja projekta primera koji se koristi u ovoj sekciji za opis razvojnog okruženja. Prvi prozor dijaloga je prikazan na slici 3.3.2.1. U okviru ovog prozora se podešava naziv projekta (**Name** polje), kao i lokacija projekta (**Location** polje). Takođe se podešava radni direktorijum projekta (**Working Directory** polje). Najčešće, lokacija projekta i radni direktorijum projekta se podešavaju na istu lokaciju. Lokacija projekta definiše lokaciju projektnog fajla koji ima .xise ekstenziju, dok lokacija radnog direktorijuma definiše gde će biti čuvani svi fajlovi koji se generišu tokom rada na projektu, pre svega fajlovi koji predstavljaju rezultat procesa kompajliranja. Uglavnom, nema potrebe razdvajati ove dve lokacije pa se najčešće i lokacija projekta i radni direktorijum projekta podešavaju na istu lokaciju, odnosno direktorijum. U **Description** polje se može uneti opis projekta, ako za tim ima potrebe. Padajući meni na dnu prozora omogućava izbor tipa top level entiteta. Opcije koje nudi ovaj padajući meni su: **HDL**, **Schematic**, **EDIF**, **NGC/NGO**. **HDL** opcija podrazumeva da je top level entitet napisan u nekom od HDL jezika poput VHDL ili Verilog jezika. **Schematic** opcija podrazumeva da je top level entitet kreiran grafički u vidu elektronske sheme. **EDIF** i **NGC/NGO** opcije podrazumevaju da je top level entitet kreiran u obliku netliste gde sam naziv opcije definiše tip netliste (EDIF, NGC ili NGO tip netliste). Napomenimo da čitav projekat ne mora da sadrži samo tip koji odgovara top level entitetu, već mogu da se koriste različiti tipovi. Na primer, top level entitet može da bude **Schematic** tipa, ali u projekat mogu da se uključe i VHDL fajlovi. Pošto se u okviru kursa obrađuje VHDL jezik, treba izabrati **HDL** opciju. Dugme **More Info** u donjem levom uglu prozora otvara odgovarajući **Help** prozor koji sadrži dodatna objašnjenja svih parametara prikazanim na prvom prozoru dijaloga za kreiranje novog projekta.



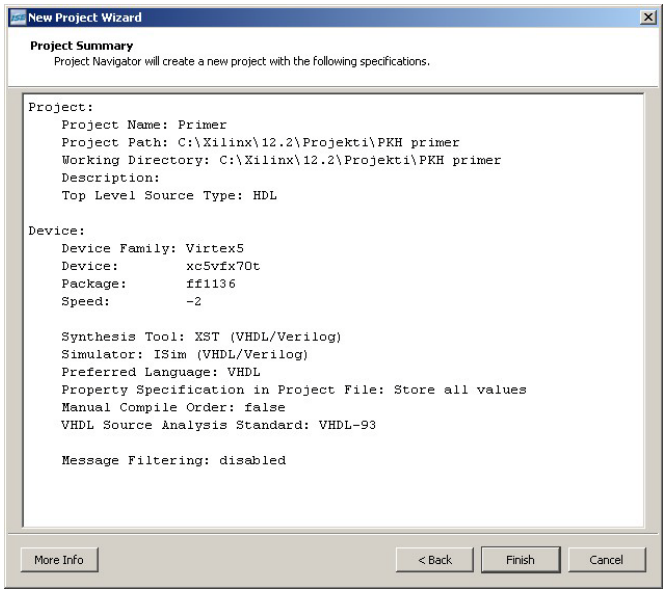
Slika 3.3.2.1. – Prozor za definisanje lokacije projekta

Sledeći prozor (slika 3.3.2.2) daje mogućnost izbora FPGA čipa, kao i selekciju osnovnih podešavanja alata, poput alata za proces sinteze dizajna i simulatora, pri čemu se ovde mogu izabrati i alati drugih proizvođača ako su na raspolaganju. Takođe, aktiviranjem opcije **Manual Compile Order** se može ručno podesiti redosled operacija procesa kompajliranja. U polju **Preffered Language** se bira poželjniji HDL jezik - VHDL ili Verilog, dok se u polju **VHDL Source Analysis Standard** podešava po kom standardu treba analizirati VHDL kod. I ovde postoji dugme **More Info** koje ima istu funkciju kao i na prethodnom prozoru. Kao što vidimo postoji i dugme **Back**, tako da se u toku dijaloga za kreiranje novog projekta uvek možemo vratiti unazad i promeniti trenutna podešavanja ukoliko to želimo.



Slika 3.3.2.2. – Prozor za izbor FPGA čipa i simulatora

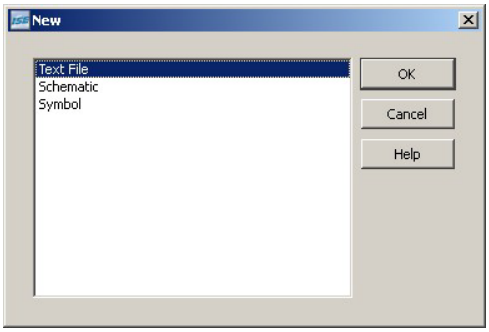
Poslednji prozor u dijalogu sadrži samo rezime podešavanja koja su urađena na prethodna dva prozora. Klikom na dugme **Finish** će biti kreiran novi projekat.



Slika 3.3.2.3. – Rezime kreiranog projekta

3.3.3. Unos dizajna

Nakon kreiranja projekta je potrebno kreirati i sam dizajn, na primer kreirati jedan ili više VHDL fajlova koji opisuju dizajn koji realizujemo tj. implementiramo. Kreiraćemo identična tri VHDL fajla (**dekadni_brojac.vhd**, **displej.vhd** i **top_level_primer.vhd**) kao i u primeru korišćenom prilikom opisa Quartus softvera. Izborom opcije **File->New** se otvara novi prozor u kom se nudi izbor tipa fajla koji želimo da kreiramo (slika 3.3.3.1).



Slika 3.3.3.1. – Izbor tipa novog fajla

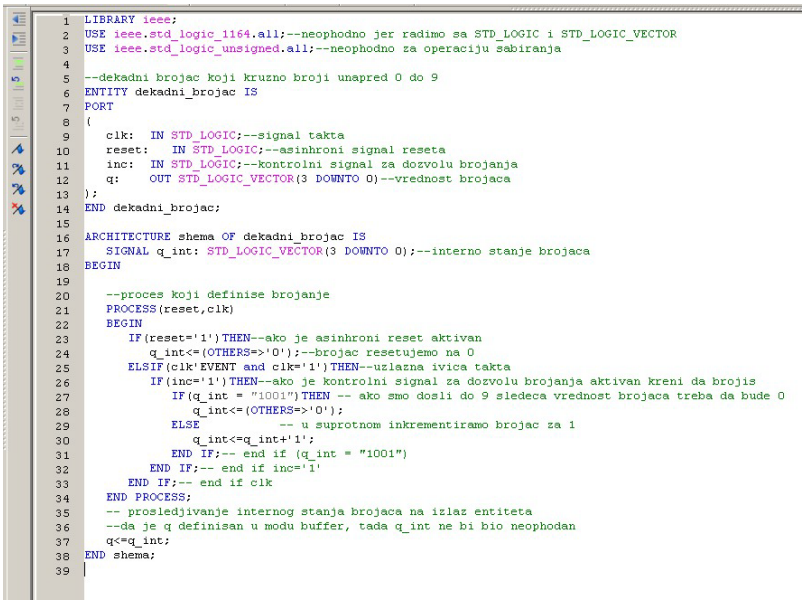
Text File opcija omogućava kreiranje fajla upotrebom tekstualnog editora i ovu opciju selektujemo za kreiranje VHDL fajla. **Schematic** opcija otvara grafički editor u kome se kreira željena shema. **Symbol** opcija otvara editor za kreiranje simbola. Simboli se koriste za grafički opis nekog modula u vidu ’crne kutije’ gde su vidljivi samo portovi modula, ali ne i unutrašnjost modula.

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;--neophodno jer radimo sa STD_LOGIC i STD_LOGIC_VECTOR
3  USE ieee.std_logic_unsigned.all;--neophodno za operaciju sabiranja
4
5  --dekadni brojac koji kruzno broji unapred 0 do 9
6  ENTITY dekadni_brojac IS
7  PORT
8  (
9     clk:  IN STD_LOGIC;--signal takta
10    reset:   IN STD_LOGIC;--asinhroni signal reseta
11    inc:  IN STD_LOGIC;--kontrolni signal za dozvolu brojanja
12    q:    OUT STD_LOGIC_VECTOR(3 DOWNTO 0)--vrednost brojaca
13 );
14 END dekadni_brojac;
15
16 ARCHITECTURE shema OF dekadni_brojac IS
17    SIGNAL q_int: STD_LOGIC_VECTOR(3 DOWNTO 0);--interno stanje brojaca
18 BEGIN
19
20    --proces koji definise brojanje
21    PROCESS(reset,clk)
22    BEGIN
23       IF(reset='1')THEN--ako je asinhroni reset aktivan
24          q_int<=(OTHERS=>'0');--brojac resetujemo na 0
25       ELSIF(clk'EVENT and clk='1')THEN--uzlazna ivica takta
26          IF(inc='1')THEN--ako je kontrolni signal za dozvolu brojanja aktivan kreni da brojis
27             IF(q_int = "1001")THEN -- ako smo dosli do 9 sledeca vrednost brojaca treba da bude 0
28                q_int<=(OTHERS=>'0');
29             ELSE           -- u suprotnom inkrementiramo brojac za 1
30                q_int<=q_int+'1';
31             END IF;-- end if (q_int = "1001")
32          END IF;-- end if inc='1'
33       END IF;-- end if clk
34    END PROCESS;
35    -- prosledjivanje internog stanja brojaca na izlaz entiteta
36    --da je q definisan u modu buffer, tada q_int ne bi bio neophodan
37    q<=q_int;
38 END shema;
39
```

Slika 3.3.3.2. – Editor u centralnom delu glavnog prozora

Prikaz tekstualnog editora je dat na slici 3.3.3.2 kada je kopiran VHDL kod dekadnog brojača dat u sekciji 3.2.3. Može se uočiti da je tekst neobeležen tj. nisu obeležene ključne reči VHDL jezika. Razlog je što fajl još nije snimljen i samim tim nije definisan koji jezik je u pitanju, pa editor tretira kod kao običan tekst. Kada snimimo ovaj fajl kao VHDL fajl, editor će onda prepoznavati ključne reči VHDL koda, kao i komentare i u skladu sa tim će ih vidljivo naznačiti u editoru, kao na slici 3.3.3.3. Otuda je najbolje odmah na početku snimiti fajl sa odgovarajućim nazivom i .vhd ekstenzijom, pa tek onda početi pisati kod u editoru jer će tada

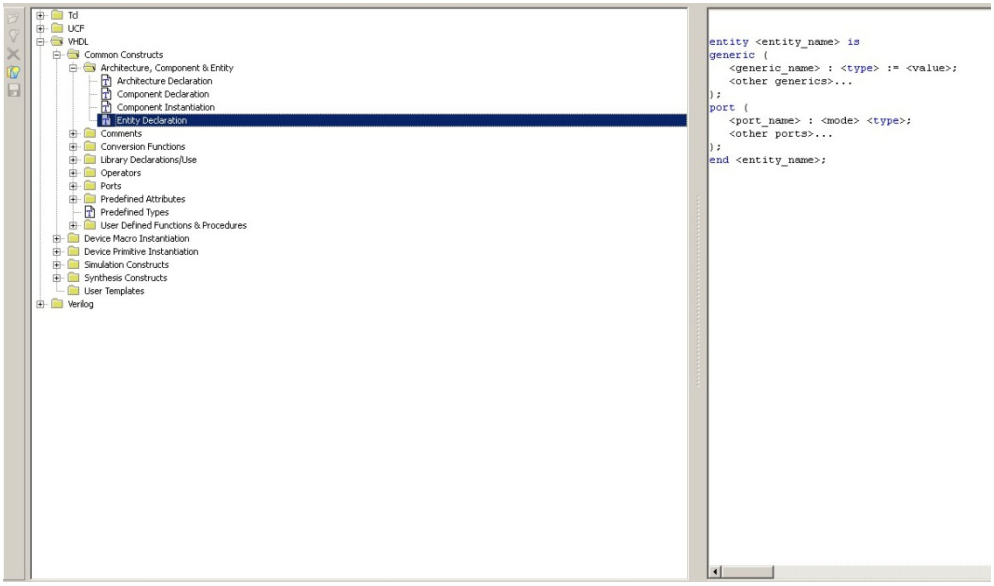
ključne reči i komentari jasno da se istaknu što čini pisanje koda znatno lakšim. U koloni sa leve strane se nalaze ikonice za editovanje. Sve opcije koje su predstavljene ikonicama u koloni sa leve strane, pa i dodatne opcije za editovanje se nalaze u glavnom meniju u okviru **Edit** grupe. U okviru **View** grupe opcija glavnog menija se mogu koristiti i opcije za prikaz koda, poput prikaza rednih brojeva linija, prikaza markera za kraj linije i sl.



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;--neophodno jer radimo sa STD_LOGIC i STD_LOGIC_VECTOR
USE ieee.std_logic_unsigned.all;--neophodno za operaciju sabiranja

--dekadni brojac koji kruzno broji unapred 0 do 9
ENTITY dekadni_brojac IS
PORT
(
   clk:  IN STD_LOGIC;--signal takta
   reset:   IN STD_LOGIC;--asinhroni signal reseta
   inc:  IN STD_LOGIC;--kontrolni signal za dozvolu brojanja
   q:    OUT STD_LOGIC_VECTOR(3 DOWNTO 0)--vrednost brojaca
);
END dekadni_brojac;

ARCHITECTURE shema OF dekadni_brojac IS
   SIGNAL q_int: STD_LOGIC_VECTOR(3 DOWNTO 0);--interno stanje brojaca
BEGIN

   --proces koji definise brojanje
   PROCESS(reset,clk)
   BEGIN
      IF(reset='1')THEN--ako je asinhroni reset aktivan
         q_int<=(OTHERS=>'0');--brojac resetujemo na 0
      ELSIF(clk'EVENT and clk='1')THEN--uzlazna ivica takta
         IF(inc='1')THEN--ako je kontrolni signal za dozvolu brojanja aktivan kreni da brojis
            IF(q_int = "1001")THEN -- ako smo dosli do 9 sledeca vrednost brojaca treba da bude 0
               q_int<=(OTHERS=>'0');
            ELSE                   -- u suprotnom inkrementiramo brojac za 1
               q_int<=q_int+'1';
            END IF;-- end if (q_int = "1001")
         END IF;-- end if inc='1'
      END IF;-- end if clk
   END PROCESS;
   -- prosledjivanje internog stanja brojaca na izlaz entiteta
   --da je q definisan u modu buffer, tada q_int ne bi bio neophodan
   q<=q_int;
END shema;
```

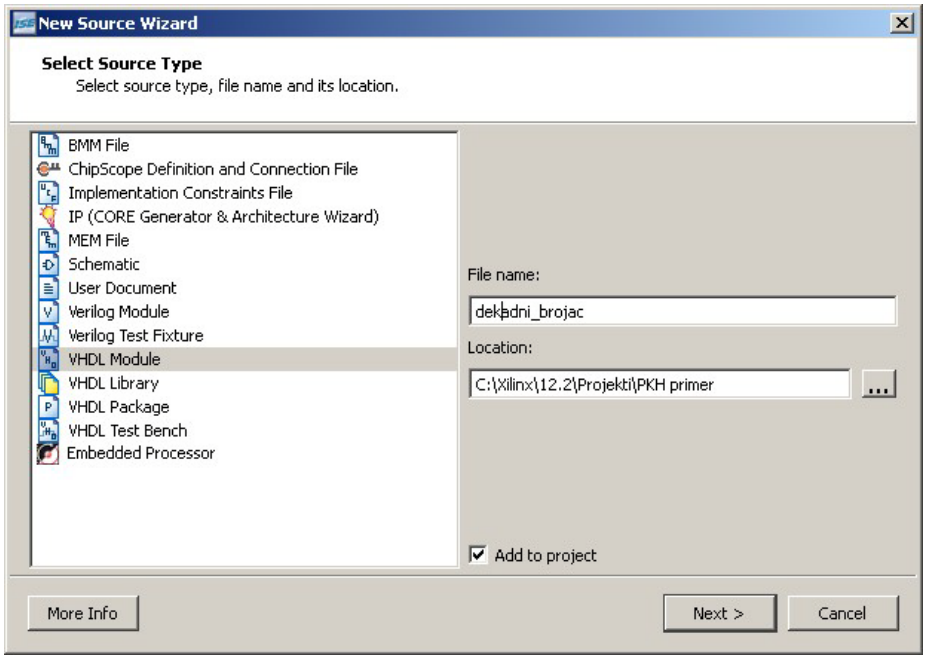
Slika 3.3.3.3. – Prikaz koda u editoru kada je editoru poznat jezik kojim je pisan kod

I ovde postoji opcija upotrebe templejta za programske jezike kao i kod Quartus softvera. Opcija **Edit->Language Templates** otvara prozor za brauzovanje kroz templejte jezika čime se na lak način mogu selektovati i kopirati željene konstrukcije i deklaracije kako VHDL jezika tako i drugih jezika (slika 3.3.3.4). Takođe, desnim klikom miša unutar editora se dobija meni u okviru koga se nalaze i opcije za komentarisanje/otkomentarisanje selektovanih linija koda, kao i aktivaciju/dekativaciju prikaza rednih brojeva linija koda.



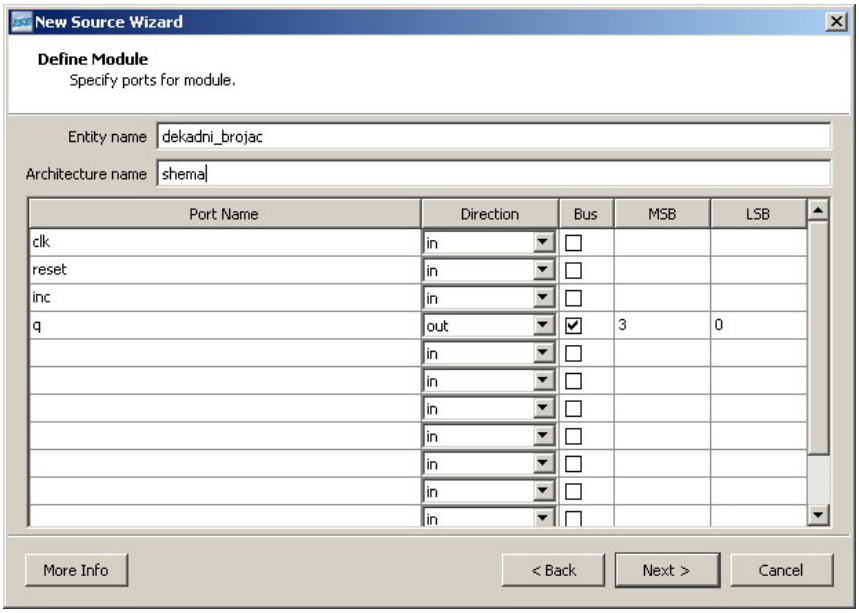
Slika 3.3.3.4. – Prozor za izbor templejta

Drugi način za kreiranje fajla je izbor opcije **Project->New Source** iz glavnog menija čime se otvara prozor za izbor tipa izvorišnog fajla koji želimo da kreiramo. Ponuđeno je više tipova fajlova među kojima i **VHDL Module** fajl koji omogućava kreiranje novog VHDL entiteta. Ovaj tip fajla bi izabrali za kreiranje svakog od tri fajla projekta primera (**dekadni_brojac.vhd**, **displej.vhd** i **top_level_primer.vhd**). Opcija **Add to project** je po difoltu aktivirana čime se kreirani fajl automatski dodaje u projekat, a ako ne želimo da fajl bude automatski dodat u projekat treba deaktivirati ovu opciju. U **File name** polje treba uneti naziv fajla, a u **Location** polje lokaciju gde treba snimiti kreirani fajl. Napomenimo da se izborom opcije **VHDL Package** kreira .vhd fajl koji po difoltu sadrži templejt VHDL paketa. Opcija **VHDL Test Bench** kreira .vhd fajl koji po difoltu sadrži testbenč (*testbench*) templejt za potrebe simulacije. U okviru ove opcije se na početku otvara dodatni prozor za selekciju entiteta koji želi da se testira i potom se tek kreira .vhd fajl koji sadrži templejt testbenča u koji je ubačen i entitet koji se testira u vidu komponente. Sam princip testbenča će biti objašnjen kasnije u okviru sekcije koja opisuje proces funkcionalne simulacije. U principu, svejedno je koju od ove tri VHDL opcije izaberemo, jer u tekstualnom editoru možemo da kreiramo sadržaj VHDL fajla koji želimo. Na primer, bez obzira što smo izabrali **VHDL Module** opciju, u tekstualnom editoru možemo da modifikujemo kreirani fajl tako da on predstavlja VHDL paket ili VHDL testbenč. Odnosno, bilo koja od tri opcije da se izabere, u tekstualnom editoru se može modifikovati kod tako da se dobiju preostale dve opcije. Svrha navedene tri opcije je da omoguće dizajneru da, uz upotrebu templejta koji se po difoltu ubacuju u sadržaj kreiranog fajla, brže napiše željeni kod.



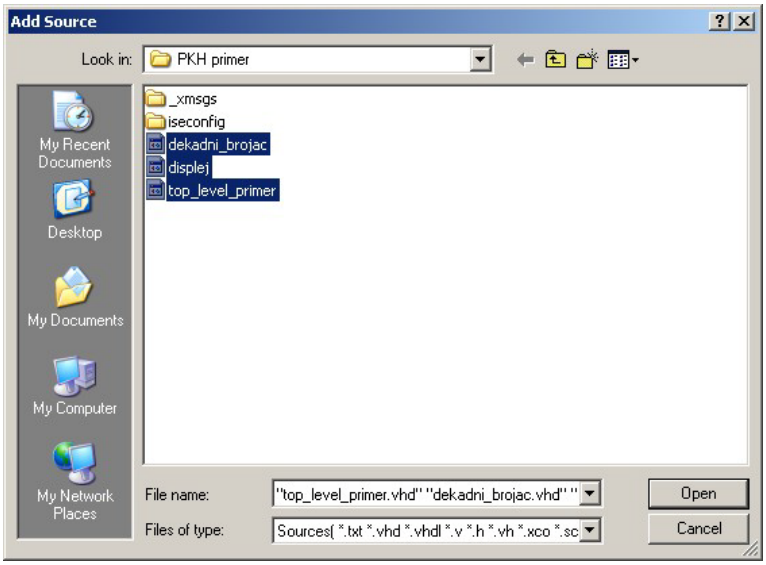
Slika 3.3.3.5. – Izbor tipa izvorišnog fajla koji kreiramo

Nakon izbora tipa izvorišnog fajla i unosa podataka o nazivu fajla i lokaciji, eventualno se otvara dodatni prozor za unos dodatnih informacija. Na primer, u slučaju izabrane opcije **VHDL Module** otvara se prozor za unos naziva entiteta, naziva arhitekture entiteta, kao i unos portova entiteta (slika 3.3.3.6).



Slika 3.3.3.6. – Unos naziva entiteta, naziva arhitekture i portova entiteta

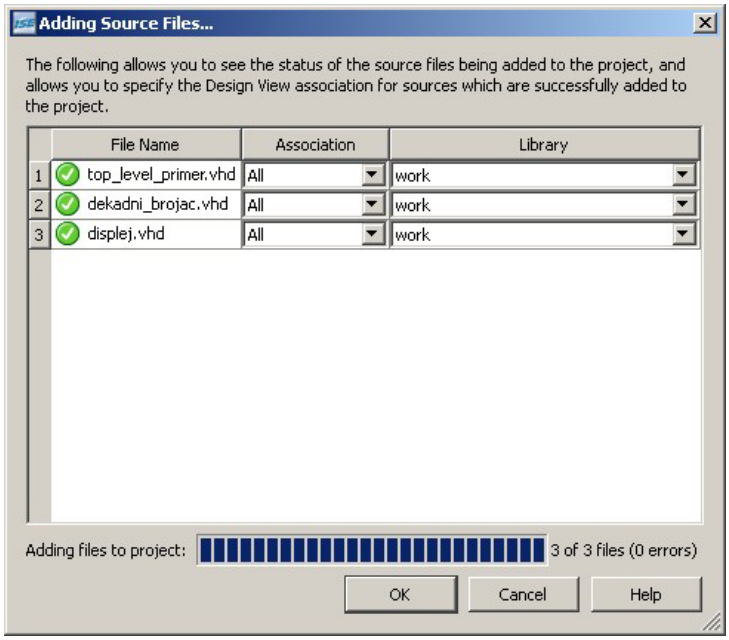
Nakon unosa naziva i portova entiteta, otvara se prozor sa rezimeom unetih parametara i nakon toga se kreirani fajl otvara u tekstualnom editoru koji smo već opisali. Sve unete informacije se mogu izmeniti u okviru tekst editora, tako da nije suštinski bitno tačno navesti podatke u okviru prozora prikazanog na slici 3.3.3.6. Svejedno je koji metod će se koristiti za kreiranje VHDL fajlova. Oba metoda na kraju koriste isti tekstualni editor. Osnovna prednost drugog metoda (**New Source**) je što ima opciju da se kreirani fajl automatski doda u projekat.



Slika 3.3.3.7. – Selekcija fajlova koje uključujemo u projekat

Nakon što smo kreirali tri VHDL fajla, treba ih uključiti u projekat ako nisu već uključeni u projekat. Opcijom **Project->Add Source** otvaramo klasičan brauzer prozor čijom upotrebom treba da selektujemo fajlove koje želimo da uključimo u projekat (slika 3.3.3.7). Nakon klika na dugme **Open** se otvara rezime uključenih fajlova - slika 3.3.3.8. Ukoliko ima neki problem sa dodavanjem selektovanog fajla u projekat to će biti označeno crvenom oznakom kod imena fajla. **Association** polje omogućava podešavanje upotrebe uključenog fajla, odnosno u okviru koje

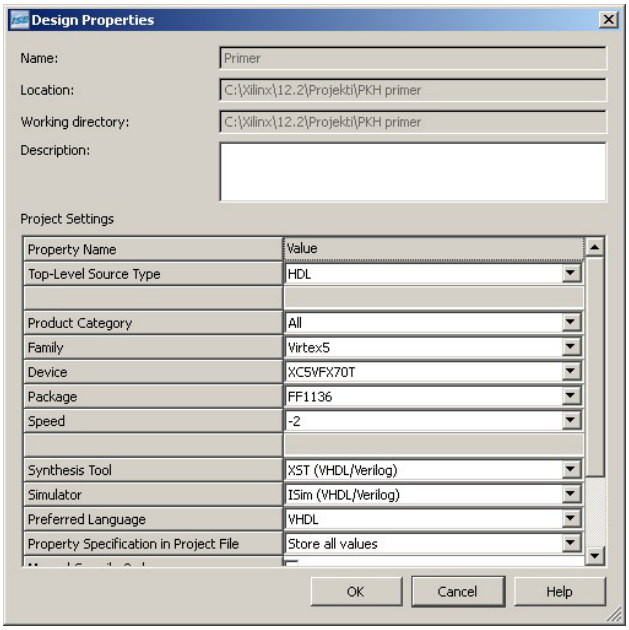
View opcije u strukturi projekta će fajl biti prikazan (**Implementation** i/ili **Simulation**). Opcije koje su u ponudi su **All**, **Implementation**, **Simulation** i **None**. **All** opcija podrazumeva da se fajl koristi i u implementaciji (generisanju konfiguracionog fajla) i u simulaciji. Ova opcija bi trebala da se podesi za sve VHDL fajlove koji predstavljaju deo implementiranog dizajna, jer se ti fajlovi, odnosno entiteti koje oni predstavljaju po pravilu trebaju proveriti u okviru simulacije. **Implementation** opcija podrazumeva da se fajl koristi samo u okviru implementacije i ova opcija se koristi za pomoćne fajlove koji sadrže informacije vezane za sam FPGA čip, poput definicije pinova na koje će biti vezani portovi top level entiteta, maksimalno dozovoljenih kašnjenja dizajna i sl. **Simulation** opcija podrazumeva da se fajl koristi samo u procesu simulacije i ona se podešava za testbenč fajlove koji predstavljaju top level celinu za simulaciju, za simulacione modele eksternih čipova koji se koriste u simulaciji, za entitete koji se koriste samo u simulaciji i sl. **None** opcija označava da se fajlu trenutno ne dodeljuje nijedna opcija i fajl je samo formalno uključen u projekat. Treba izbegavati upotrebu ove opcije. Izmena **Association** vrednosti je moguća i kasnije, nakon što je fajl dodat u projekat i to tako što se selektuje željeni fajl u strukturi projekta ili listi fajlova (tab **Files** ispod toka projekta), a onda se u glavnom meniju izabere opcija **Source->Source Properties** čime se otvara prozor u kome može da se promeni **Association** vrednost (**Source Properties** opcija je dostupna i u meniju koji se otvara desnim klikom na željeni fajl u strukturi projekta ili listi fajlova). **Library** daje mogućnost izbora biblioteke u okviru koje će biti uključen dodati fajl. Po difoltu je u pitanju **work** biblioteka projekta, ali može da se selektuje i neka druga ako se to želi. Nakon dodavanja fajlova, ISE softver automatski prepoznaje top level entitet i u strukturi projekta prikazuje hijerarhiju fajlova u projektu. Ako se želi selektovati neki drugi fajl (entitet) kao top level entitet, tada se desnim klikom na željeni entitet izabere opcija **Set as Top Module** u meniju koji se otvori na desni klik miša. Ista opcija se može selektovati i u glavnom meniju u **Source** grupi opcija, pri čemu je pre izbora ove opcije potrebno selektovati željeni fajl u strukturi projekta. U okviru **Source** grupe opcija postoji i opcija **Remove** za uklanjanje selektovanog fajla iz projekta. Još jednom napomenimo da se fajl može selektovati u strukturi projekta, ali i u listi fajlova kad je selektovan tab **Files** ispod toka projekta.



Slika 3.3.3.8. – Rezime i podešavanja selektovanih fajlova

3.3.4. Podešavanje opcija projekta i kompajlera

Nakon što smo kreirali dizajn i definisali top level entitet, potrebno je pokrenuti i proces kompajliranja. Međutim, pre pokretanja procesa kompajliranja neophodno je podesiti parametre kompajliranja, ali i eventualno promeniti parametre projekta (na primer promeniti FPGA čip). Za razliku od Quartus paketa gde su podešavanja projekta bila uglavnom okupljena na jednom mestu, kod ISE paketa to nije slučaj. Radi ostvarivanja konzistentnosti opisa, objašnjenja podešavanja opcija projekta i kompajlera ISE softverskog paketa će uglavnom pratiti redosled kojim su opisana podešavanja kod Quartus softverskog paketa u sekciji 3.2.4.

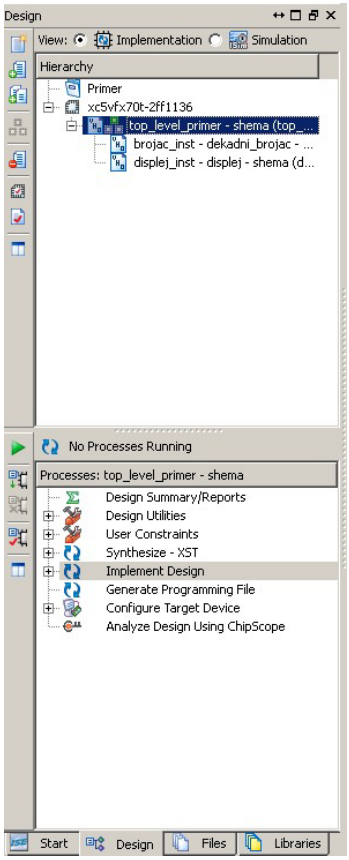


Slika 3.3.4.1. – Prozor za promenu osnovnih podešavanja projekta

Izborom opcije **Project->Design Properties** iz glavnog menija otvara se prozor koji omogućava promenu podešavanja projekta definisanih prilikom kreiranja projekta, poput izbora čipa, alata za simulaciju, tipa top level entiteta i sl. Naziv projekta, kao i lokacija i radni direktorijum projekta ne mogu da se menjaju kao što se vidi sa slike 3.3.4.1. Napomenimo da u slučaju da su i radni direktorijum i lokacija projekta postavljeni na isti direktorijum (što je uglavnom i slučaj), direktorijum koji sadrži projekat se može kopirati na proizvoljnu lokaciju, ili se može promeniti ime direktorijuma. Po otvaranju projekta na novoj lokaciji (u to spada i slučaj kada se samo promeni naziv direktorijuma koji sadrži projekat), ISE će automatski promeniti lokaciju i radni direktorijum na novi direktorijum koji će biti prikazan u odgovarajućim poljima prozora za promenu osnovnih podešavanja projekta prikazanog na slici 3.3.4.1.

Izborom opcije **Project->Add Source** iz glavnog menija se otvara prozor za dodavanje novih fajlova u projekat. Ovu opciju smo već opisali u prethodnoj sekciji 3.3.3. Ova opcija se takođe može pozvati i desnim klikom na polje strukture projekta ili desnim klikom na polje koje sadrži listu fajlova uključenih u projekat (kada je izabran tab **Files** ispod toka projekta). Uklanjanje fajlova se postiže selekcijom željenog fajla u strukturi projekta ili listi fajlova i pozivanjem **Remove** opcije ili desnim klikom na izabrani fajl ili izborom opcije **Source-> Remove** iz glavnog menija.

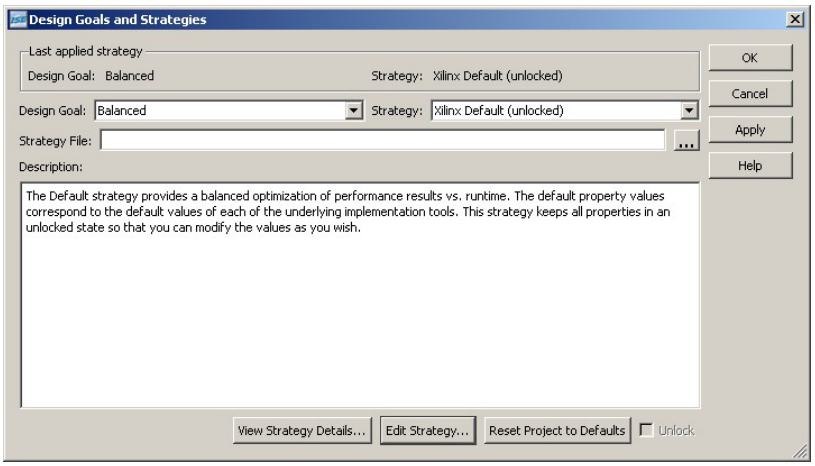
Selekcijom top level entiteta u strukturi projekta, kada je aktivan **Implementation View**, u toku projekta se dobija lista koraka procesa kompajliranja (slika 3.3.4.2). Kao što vidimo sa slike 3.3.4.2, postoje tri osnovna koraka procesa kompajliranja - proces analize i sinteze (**Synthetize**), proces postavljanja i rutiranja (**Implement Design**) i generisanje konfiguracionog fajla FPGA čipa (**Generate Programming File**). Meni za podešavanja parametara svakog navedenog koraka se dobijaju tako što se u toku projekta selektuje željeni korak procesa kompajliranja i potom iz glavnog menija izabere opcija **Process->Process Properties**. Drugi način je desni klik na željeni korak procesa kompajliranja i potom izbor opcije **Process Properties** u meniju koji se otvori na desni klik miša.



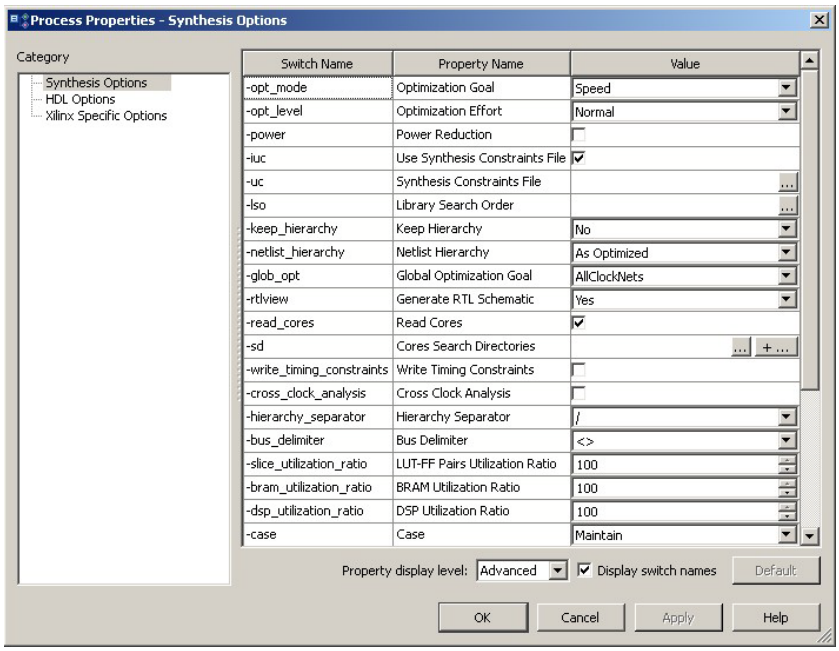
Slika 3.3.4.2. – Lista koraka procesa kompajliranja

Opcija **Project->Design Goals & Strategies** iz glavnog menija daje mogućnost podešavanja cilja optimizacionog algoritma koji se koristi u okviru procesa kompajliranja (slika 3.3.4.3). Cilj može biti da dizajn zauzme što manje resursa na FPGA čipu, da ostvari što veću maksimalnu frekvenciju dizajna, da što kraće traje proces kompajliranja, da rezultujući dizajn što manje troši snage (cilj je energetski efikasan dizajn) i balansirani dizajn koji pokušava da nađe rešenje koje će pokušati u što većoj meri da zadovolji prethodno navedena četiri kriterijuma optimizacije. Ove opcije se biraju u padajućem meniju **Design Goal**. Svaka od opcija (tj. kriterijuma optimizacije) zaključava određeni skup parametara u podešavanjima koraka procesa kompajliranja. Koji parametri su u pitanju se može videti klikom na dugme **View Strategy Details** u dnu prozora. **Edit Strategy** dugme u dnu prozora omogućava da se modifikuju parametri strategije. Modifikovana strategija se može sačuvati radi upotrebe u drugim projektima. **Strategy File** polje omogućava poziv korisnički definisane strategije. Očigledno,

upotrebom opcije **Design Goals & Strategies** se efikasno može odjednom postaviti više parametara procesa kompajliranja, a takođe ako se definiše i sačuva korisnički definisana strategija, njenim učitavanjem se lako podešavaju svi parametri procesa kompajliranja odjednom u projektima u kojima nam odgovara takva strategija.



Slika 3.3.4.3. – Prozor za podešavanje cilja optimizacije procesa kompajliranja



Slika 3.3.4.4. – Prozor za podešavanje parametara procesa analize i sinteze

Design Utilities skup opcija sadrži mogućnost generisanja simbola za dati VHDL entitet (**Create Schematic Symbol**), kao i opciju prikaza templejta instanciranja u vidu komponente selektovanog entiteta (**View HDL Instantiation Template**). Ovaj skup opcija je vidljiv za bilo koji selektovani entitet u strukturi projekta.

User Constraints skup opcija omogućava kreiranje korisničkih ograničenja koja se koriste u procesu postavljanja i rutiranja poput dodele pinova portovima top level entiteta, definisanja željenih frekvencija za signale takta u dizajnu i dr. Ove opcije će biti objašnjene u kasnijim sekcijama.

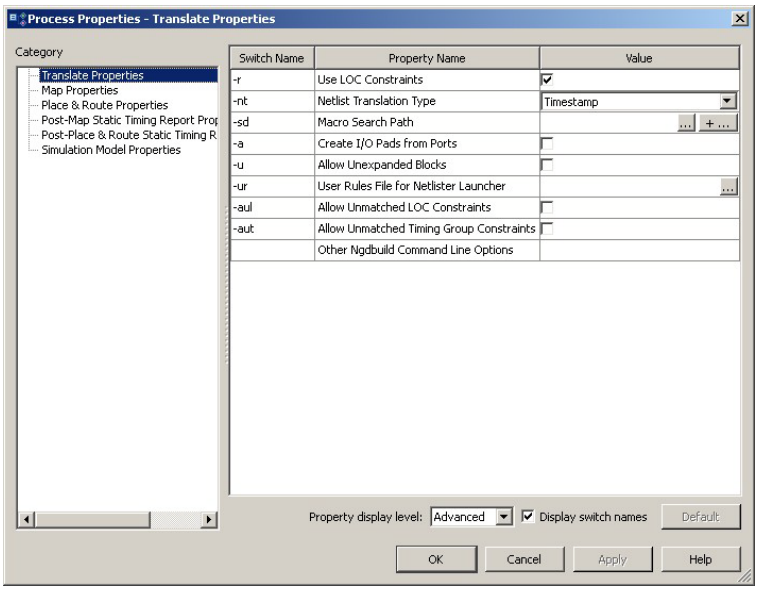
Selektovanjem **Synthesize** koraka u toku kompajliranja i potom izborom opcije **Process ->Process Properties** iz glavnog menija otvara se prozor za podešavanje procesa analize i sinteze (slika 3.3.4.4). Postoje tri grupe parametara u meniju sa leve strane čijim izborom se otvara lista parametara selektovane grupe u desnom delu prozora. **Synthesis Options** grupa se odnosi na opšta podešavanja procesa analize i sinteze, poput cilja optimizacije (**Speed** ili **Area**), nivo truda optimizacionog algoritma da se dođe do što boljeg rešenja, i dr. Klikom na **Help** dugme u donjem delu prozora, otvara se prozor sa objašnjenjima parametara prikazanih u desnom delu prozora (ovo važi za sve tri grupe parametara). **HDL Options** grupa se odnosi na definisanje ponašanja i akcije kompajlera na razne slučajeve u HDL kodu. Na primer, ako se uoči da u kodu postoji konačni automat, **FSM Encoding Algorithm** parametar definiše način kodiranja stanja konačnog automata. **Xilinx Specific Options** grupa sadrži parametre koji uzimaju u obzir i tehnologiju Xilinx-ovih čipova. Neki od parametara koji su ovde ponuđeni su automatsko dodavanje ulaznih/izlaznih bafera na portove top level entiteta ukoliko ih korisnik nije uneo u dizajn, definisanje maksimalnog *fan-out* parametra, dupliranje registara i dr. Za svaku grupu u donjem delu prozora postoji padajući meni koji definiše prikaz parametara. **Advanced** opcija daje prikaz svih parametara selektovane grupe. **Standard** opcija daje prikaz najvažnijih parametara selektovane grupe.

U okviru **Synthesize** koraka u toku kompajliranja se nalazi i opcija **Generate Post-Synthesis Simulation Model** koja generiše simulacioni modul dizajna koji se može posle uključiti u proces simulacije. Generisani simulacioni modul uzima u obzir krajnji rezultat procesa analize i sinteze dizajna. Podešavanja ove opcije se dobijaju njenom selekcijom i potom izborom **Process->Process Properties** iz glavnog menija čime se otvara prozor za podešavanje parametara ove opcije. Otvoreni prozor je veoma sličan prozoru prikazanom na slici 3.3.4.4, uz razliku što ne postoji meni sa leve strane jer postoji samo jedna grupa parametara. I ovde postoji **Help** dugme kojim se otvara prozor sa objašnjenjima svih navedenih parametara, a takođe postoji i padajući meni sa izborom **Advanced** ili **Standard** prikaza parametara. Jedan od parametara koji se podešava je i jezik u kom će biti generisan simulacioni modul (VHDL ili Verilog).

Takođe, u okviru **Synthesize** koraka u toku kompajliranja se nalazi i opcija **Check Syntax** kojom se proverava ispravnost sintakse top level entiteta. Ova opcija je dostupna i za ostale fajlove projekta. Naime, kada se selektuje fajl koji ne predstavlja top level entitet, u toku projekta se prikazuje skraćeni meni gde je ponuđena i opcija **Check Syntax**.

Selektovanjem **Implement Design** koraka u toku kompajliranja i potom izborom opcije **Process->Process Properties** iz glavnog menija otvara se prozor za podešavanje procesa postavljanja i rutiranja, kao i procesa vremenske analize (slika 3.3.4.5). Sa leve strane prozora se nalazi lista koraka procesa postavljanja i rutiranja, a sa desne strane se nalazi lista parametara selektovanog koraka. I ovde postoji **Help** dugme kojim se otvara prozor sa objašnjenjima svih navedenih parametara, a takođe postoji i padajući meni sa izborom **Advanced** ili **Standard** prikaza parametara. Korak prevođenja (**Translate**) podrazumeva korak u kome se uzimaju sve netliste generisane u procesu analize i sinteze, potom se te netliste spajaju, zatim se uzimaju u obzir eventualna korisnička ograničenja (na primer dodela pinova) i generiše se .ngd fajl koji se potom koristi u procesu mapiranja na selektovani Xilinx čip (koji je podešen u projektu - slika 3.3.4.1). Parametri koraka prevođenja se nalaze pod grupom **Translate Properties**. Korak mapiranja (**Map**) potom uzima rezultat koraka prevođenja i mapira dizajn na izabrani FPGA čip, tačnije mapira na generičke delove FPGA čipa (logičke elemente tj. CLB blokove, interne

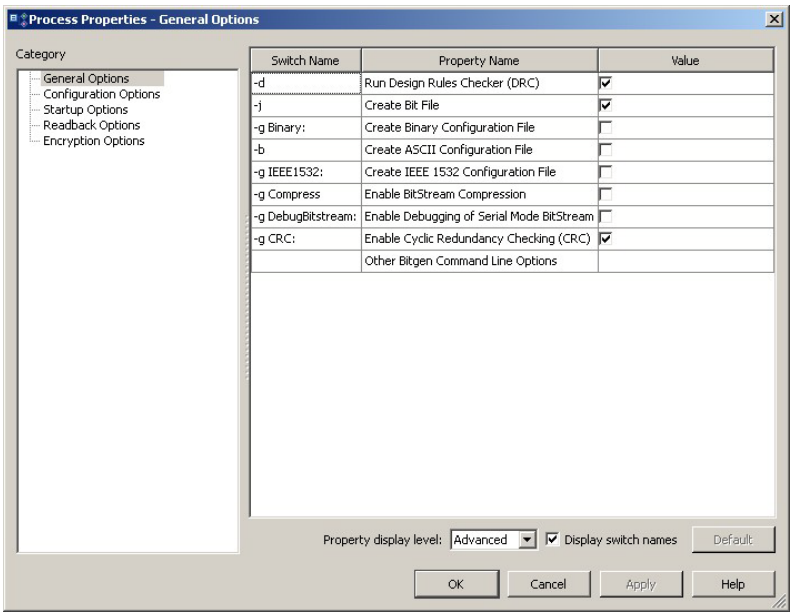
memorije i dr.), ali ne uzimajući u obzir raspored delova FPGA čipa. Cilj mapiranja je što optimalniji raspored dizajna po delovima FPGA čipa, na primer da se zauzme što manje resursa. Ali, ovde još nisu delovi dizajna raspoređeni po FPGA čipu, tačnije po konkretnim lokacijama FPGA čipa. Parametri koraka mapiranja se nalaze pod grupom **Map Properties**. Rezultat koraka mapiranja predstavlja ulazni podatak za korak postavljanja i rutiranja (**Place & Route**), koji sada razmešta dizajn na konkretne lokacije u FPGA čipu u cilju postizanja što manjeg kašnjenja dizajna tj. što efikasnijeg rasporeda dizajna. Na osnovu rezultata koraka postavljanja i rutiranja će biti posle generisan konfiguracioni fajl za programiranje FPGA čipa. Parametri koraka postavljanja i rutiranja se nalaze pod grupom **Place & Route Properties**. Kao što smo već naveli, u okviru **Implement Design** opcije se nalazi i proces vremenske analize. Postoje dva nivoa vremenske analize, nakon koraka mapiranja (**Post-Map Static Timing**) i nakon koraka postavljanja i rutiranja (**Post-Place & Route Static Timing**). Vremenska analiza nakon mapiranja daje uvid u tzv. logička kašnjenja gde se pod logičkim kašnjenjem podrazumeva broj nivoa kombinacione logike. Pošto mapiranje još nije razmestilo dizajn po konkretnim lokacijama FPGA čipa, onda se kašnjenje može samo logički proceniti u smislu da što je kombinaciona logika na nekoj putanji sa većim brojem nivoa to će biti veće i kašnjenje te putanje. Ova analiza je zgodna za detektovanje kritičnih putanja i njihovih delova, pa onda u slučaju da dizajn ne zadovoljava vremenska ograničenja možemo da izvršimo korekciju dizajna i da smanjimo broj nivoa kombinacione logike kritične putanje. Vremenska analiza nakon postavljanja i rutiranja daje uvid u realna kašnjenja dizajna jer je sada dizajn razmešten na konkretne lokacije FPGA čipa, pa je moguće i precizno proceniti kašnjenje svake putanje i utvrditi da li dizajn zadovoljava postavljena vremenska ograničenja ili ne. Podešavanja za ove dve vremenske analize se nalaze pod grupama **Post-Map Static Timing Report Properties** i **Post-Place & Route Static Timing Report Properties**. U sva tri koraka procesa postavljanja i rutiranja može da se generiše simulacioni modul dizajna nakon izvršavanja odgovarajućeg koraka. U **Simulation Model Properties** grupi se podešavaju parametri generisanja simulacionog mudula.



Slika 3.3.4.5. – Prozor za podešavanje parametara procesa postavljanja i rutiranja

U okviru **Implement Design** opcije su vidljivi svi koraci procesa postavljanja i rutiranja (**Translate**, **Map** i **Place & Route**), pri čemu se selekcijom bilo kog od tih koraka i potom izborom opcije **Process->Process Properties** iz glavnog menija otvara prozor sa podešavanjem

parametara selektovanog koraka, pri čemu se u prozoru ne pojavljuje meni sa leve strane jer se prikazuju samo parametri selektovanog koraka. Isto važi i za navedena podešavanja vremenske analize, kao i simulacionih modula gde se na isti način otvara njihov prozor samo što je pre toga potrebno selektovati odgovarajuću podopciju u okviru **Implement Design** opcije.



Slika 3.3.4.6. – Prozor za podešavanje parametara procesa generisanja konfiguracionog fajla

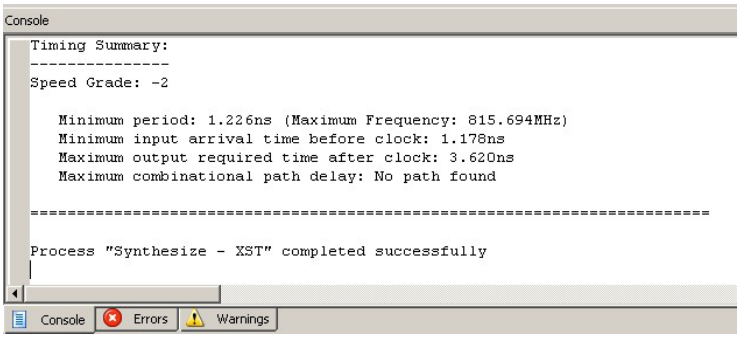
Selektovanjem **Generate Programming File** koraka u toku kompajliranja i potom izborom opcije **Process->Process Properties** iz glavnog menija otvara se prozor za podešavanje procesa generisanja konfiguracionog fajla za programiranje FPGA čipa (slika 3.3.4.6). **General Options** grupa parametara se odnosi na opšta podešavanja pre svega tipa generisanog konfiguracionog fajla, pri čemu se može selektovati više tipova odjednom, čime će proces generisanja konfiguracionog fajla kreirati više tipova zapisa konfiguracionog fajla. **Configuration Options** grupa parametara se odnosi na konfigurisanje specijalnih pinova, kao i nekorišćenih pinova, ali i druga podešavanja koja se odnose na konfigurisanje FPGA čipa. Konfigurisanje pinova podrazumeva da li će biti dodat *Pull Up* otpornik, *Pull Down* otpornik ili će se pin ostaviti da 'visi' (*float*). Kod nekih pinova su ponuđene samo neke od navedenih opcija. **Startup Options** grupa parametara se odnosi na podešavanja vezana za proces programiranja FPGA čipa. **Readback Options** grupa parametara se odnosi na podešavanja vezana za proveru ispravnosti upisane konfiguracije FPGA čipa nakon obavljenog programiranja FPGA čipa. **Encryption Options** grupa parametara se odnosi na podešavanja enkripcije generisanog konfiguracionog fajla ukoliko želimo da vršimo enkripciju konfiguracionog fajla. Enkripcija konfiguracionog fajla je zgodna opcija ukoliko se želimo zaštititi od slučaja da zlonamerni napadač na osnovu sadržaja PROM ili fleš memorije koja sadrži konfiguracioni fajl ili čitanja konfiguracionog fajla tokom programiranja FPGA čipa sazna strukturu dizajna spuštenog na FPGA čip.

Napomenimo još da sve opcije u toku projekta koje kod sebe imaju dve strelice povezane u krug predstavljaju akcije koje se mogu aktivirati dvostrukim klikom (ili njihovom selekcijom i potom izborom opcije **Process->Run** iz glavnog menija). Podešavanjima koja se odnose samo na željenu opciju se pristupa desnim klikom miša na tu opciju u toku projekta i potom izborom **Process Properties** u meniju koji se otvori na desni klik.

3.3.5. Analiza i sinteza

U ovoj sekciji ćemo opisati kako izvršiti analizu i sintezu dizajna, kao prvog koraka procesa kompajliranja dizajna. Kada smo uneli dizajn i definisali top level entitet, proces analize i sinteze se može pokrenuti na dva načina. Prvi način je dvostrukim klikom na opciju **Synthesize** u toku projekta. Drugi način je selektovanjem opcije **Synthesize**, a potom selekcijom opcije **Process->Run** iz glavnog menija. Važno je napomenuti da se dvostrukim klikom na bilo koju opciju iz toka projekta koja kod sebe ima dve strelice povezane u krug pokreće dotična opcija, odnosno taj korak kompajliranja. Pri tome, ukoliko neki od prethodnih koraka procesa kompajliranja nisu izvršeni, prvo će se oni izvršiti, pa tek onda selektovana opcija. Ostali koraci procesa kompajliranja iza selektovane opcije neće biti izvršeni.

U delu za ispis obaveštenja će biti ispisane odgovarajuće poruke koje prate tok kompajliranja (u ovom slučaju samo analize i sinteze) - slika 3.3.5.1. Tabovima se može filtrirati prikaz poruka tako da se prikažu samo poruke od interesa, na primer samo poruke o greškama (tab **Errors**) ili samo poruke upozorenja (tab **Warnings**). Tab **Console** daje prikaz svih poruka i obaveštenja generisanih tokom procesa kompajliranja, u ovom slučaju samo procesa analize i sinteze.

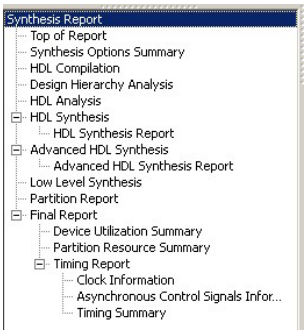


Slika 3.3.5.1. – Prikaz poruka kreiranih u toku izvršavanja procesa analize i sinteze

Slika 3.3.5.2. – Ispis sažetog izveštaja obavljenog procesa analize i sinteze

Nakon uspešno izvršene analize i sinteze u centralnom delu prozora ISE aplikacije se otvara sažeti izveštaj procesa kompajliranja prikazan na slici 3.3.5.2. Ovaj sažeti izveštaj se sastoji od četiri dela. Prvi deo daje osnovne informacije o projektu (izabrani čip, izabrana

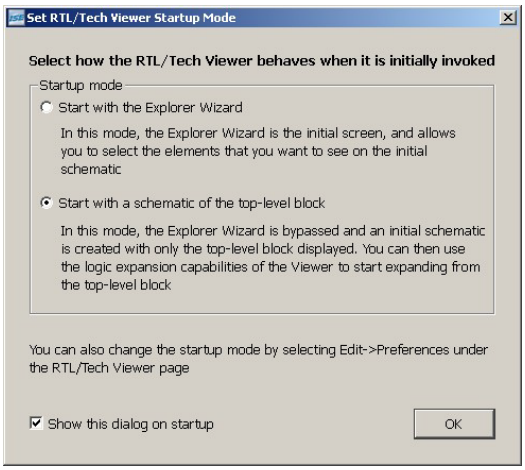
strategija za algoritam optimizacije, trenutni status toka kompajliranja). Drugi deo daje pregled upotrebljenih resursa (procenu ili realnu upotrebu, u zavisnosti od tačke gde se kompajliranje zaustavilo). U trećem delu se nalazi lista kompletnih izveštaja, pri čemu postoje samo oni izveštaji sa aktivnim hiperlinkom. Poslednji, četvrti deo daje listu dodatnih izveštaja, koji se generišu, na primer, u procesu simulacije ili kreiranja simulacionih modula dizajna. U većem delu ekrana se nakon zaustavljanja procesa kompajliranja uvek po difoltu nalazi otvoren sažeti izveštaj nakon poslednjeg izvršenog koraka, u ovom slučaju procesa analize i sinteze (polje **Implementation State** u gornjem delu sažetog izveštaja definiše dokle se stiglo u procesu kompajliranja). Važne informacije su broj zauzetih internih resursa čipa. Međutim, pošto proces analize i sinteze još nije striktno vezan za sam konkretan FPGA čip, ovde je u pitanju veoma dobra procena, koja može da se razlikuje od konačne implementacije i raspoređivanja dizajna na sam čip u okviru procesa postavljanja i rutiranja. U koloni sa leve strane se mogu otvarati izveštaji procesa kompajliranja, pri čemu izveštaji u sivoj boji nisu generisani pa samim tim ni dostupni za čitanje. Naravno, izveštaji koji nisu dostupni će biti generisani u kasnijim koracima procesa kompajliranja. Izveštaji su podeljeni i tri glavne grupe. **Design Overview** grupa daje osnovne implementacione informacije o dizajnu. **Errors and Warnings** grupa daje pregled generisanih poruka za sve korake i potkorake procesa kompajliranja. **Detailed Reports** grupa daje kompletne izveštaje koraka i potkoraka procesa kompajliranja. Ispod liste izveštaja se nalazi dodatni meni sa opcijama prikaza selektovanog izveštaja za prve dve grupe izveštaja ili opcijama za brauzovanje kroz selektovani meni za treću grupu.



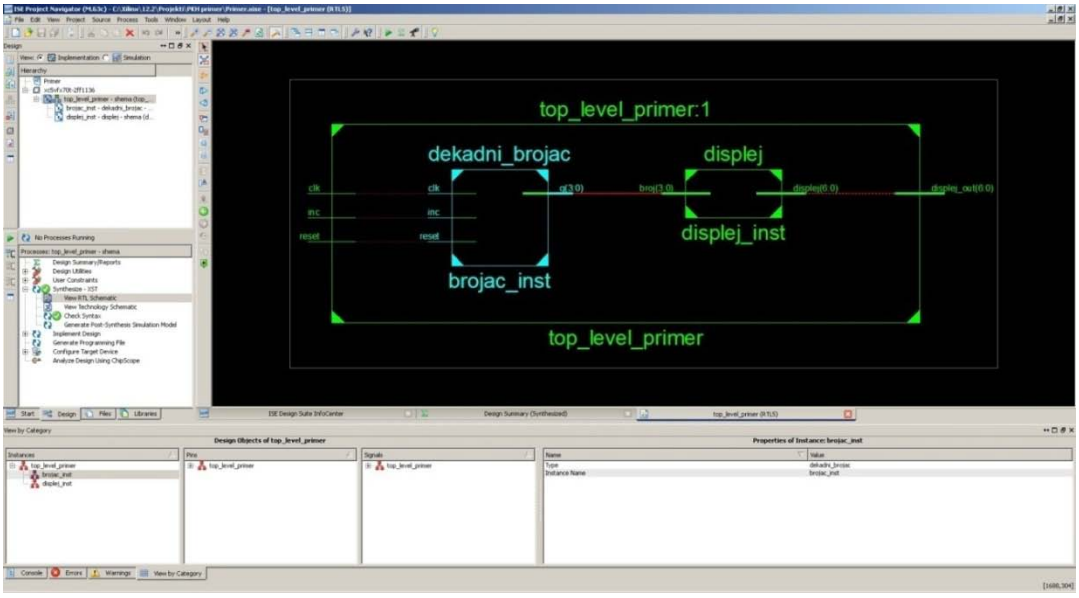
Slika 3.3.5.3. – Struktura detaljnog izveštaja procesa analize i sinteze

Struktura detaljnog izveštaja za proces analize i sinteze (**Synthesis Report**) je data na slici 3.3.5.3. Kompletan izveštaj je otvoren u centralnom delu prozora ISE aplikacije, a meni prikazan na slici 3.3.5.3 služi za efikasnije brauzovanje kroz delove izveštaja. **Synthesis Options Summary** predstavlja pregled podešavanja za proces analize i sinteze. **HDL Compilation** predstavlja pregled kompajliranih fajlova. **Design Hierarchy Analysis** daje pregled analize hijerarhijske strukture dizajna. **HDL Analysis** daje pregled analize i koje su jedinice generisane od uključenih entiteta. Praktično **HDL Compilation**, **Design Hierarchy Analysis** i **HDL Analysis** daju samo pregled toga šta je uzeto u obzir i analizirano (na primer analiza sintakse) u procesu analize dizajna. **HDL Synthesis** daje rezultat procesa sinteze gde se izveštava za svaku jedinicu dizajna (koja su generisane u procesu analize) šta je upotrebljeno za njihovu konstrukciju. U okviru **HDL Synthesis Report** podsekcije se daje zbirni izveštaj svih elemenata koji su upotrebljeni za konstrukciju svih delova dizajna. **Advanced HDL Synthesis** daje rezultat procesa napredne sinteze gde se izveštava rezultat napredne sinteze za svaku jedinicu dizajna pošto je potencijalno moglo doći do dodatne optimizacije dizajna u procesu sinteze. U okviru **Advanced HDL Synthesis Report** podsekcije se daje zbirni izveštaj svih elemenata koji su upotrebljeni za konstrukciju svih delova dizajna nakon izvršene napredne sinteze dizajna. **Low**

Level Synthesis daje rezultat konačne sinteze gde se kao rezultat generiše i optimizuje finalna netlista koja će biti korišćena u sledećem koraku procesa kompajliranja (korak prevođenja procesa postavljanja i rutiranja). **Partition Report** daje izveštaj o particijama dizajna za slučaj da dizajn sadrži particije. **Final Report** predstavlja konačni izveštaj izvršenog procesa analize i sinteze i daje pregled upotrebljenih resursa po tipovima resursa. U okviru **Device Utilization Summary** podizveštaja se daje procena iskorišćenih resursa FPGA čipa. **Partition Resource Summary** podizveštaj daje pregled iskorišćenih resursa po particijama ako one postoje u dizajnu (i ovde je u pitanju procena resursa). **Timing Report** sadrži procene vremenskih kašnjenja dizajna na FPGA čipu. Pri tome **Clock Information** daje pregled svih detektovanih taktova u dizajnu. **Asynchronous Control Signals Information** daje pregled svih asinhronih kontrolnih signala detektovanih u dizajnu. **Timing Summary** daje procenu maksimalne frekvencije dizajna, kao i pregled procenjenih najkritičnijih putanja dizajna.



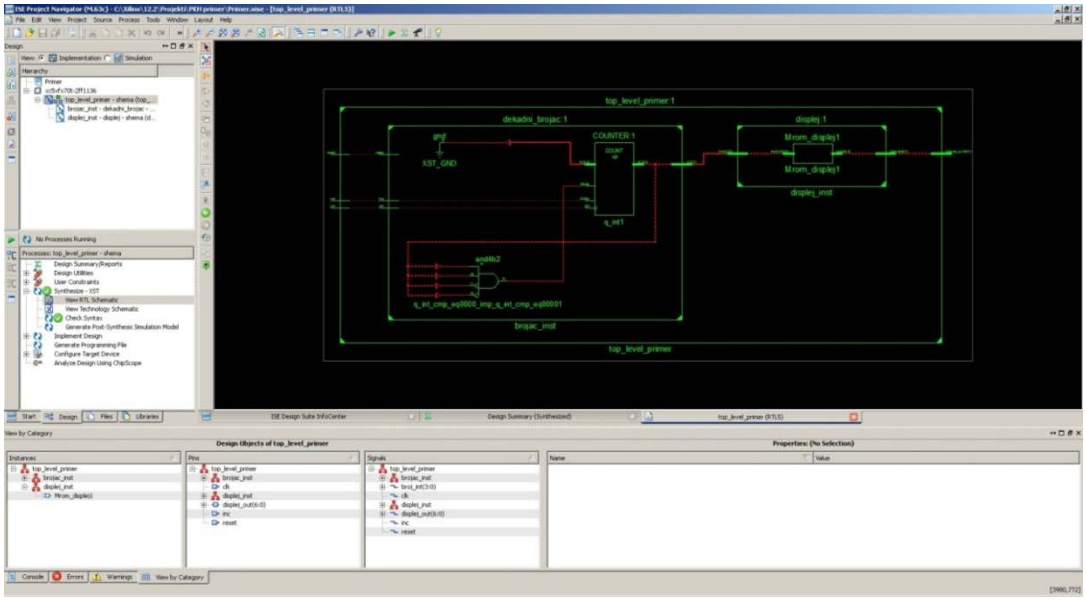
Slika 3.3.5.4. – Izbor delimičnog ili kompletnog shematskog prikaza dizajna



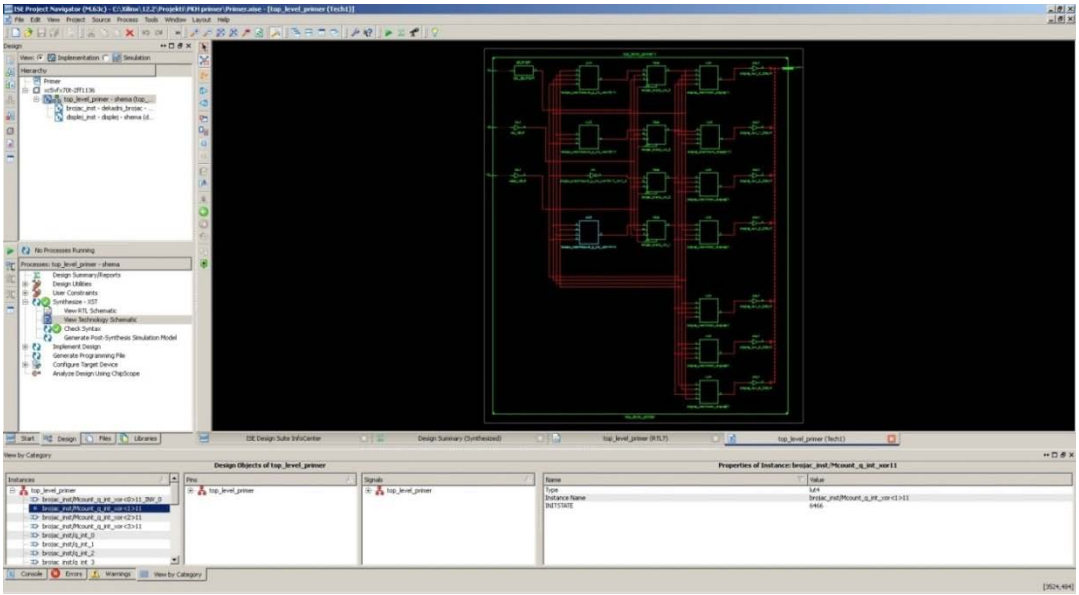
Slika 3.3.5.5. – Kompletan shematski prikaz dizajna

View RTL Schematic podopcija u okviru **Synthesize** opcije toka projekta daje prikaz sheme dizajna. Ova podopcija se pokreće dvostrukim klikom te podopcije ili njenim selektovanjem pa izborom opcije **Process->Run** glavnog menija. Na slici 3.3.5.4 je prikazan

prozor koji se otvara izborom navedene podopcije. Difolt (donja) opcija je da se kreira shema kompletnog dizajna pri čemu prikaz ide od top level dizajna. Gornja opcija otvara navigator koji omogućava selekciju delova dizajna koji želimo da shematski prikažemo, pri čemu se nakon selekcije željenih elemenata klikom na dugme **Create Schematic** kreira shematski prikaz selektovanih delova dizajna. Na slici 3.3.5.5. je prikazana shema unutrašnjeg sadržaja top level entiteta. U donjem delu prozora se nalaze delovi dizajna čijom selekcijom se obeležavaju odgovarajući elementi na shemi. Takođe, osnovne informacije selektovanog dela su prikazane u desnoj polovini donjeg dela prozora. Duplim klikom na neki blok u shemi, prikazuje se interna struktura bloka. Na taj način se može prikazati kompletna struktura dizajna (slika 3.3.5.6). Desnim klikom miša na neki blok se može otvoriti ili zatvoriti interna struktura bloka (**Show Block Contents**, **Hide Block Contents**).

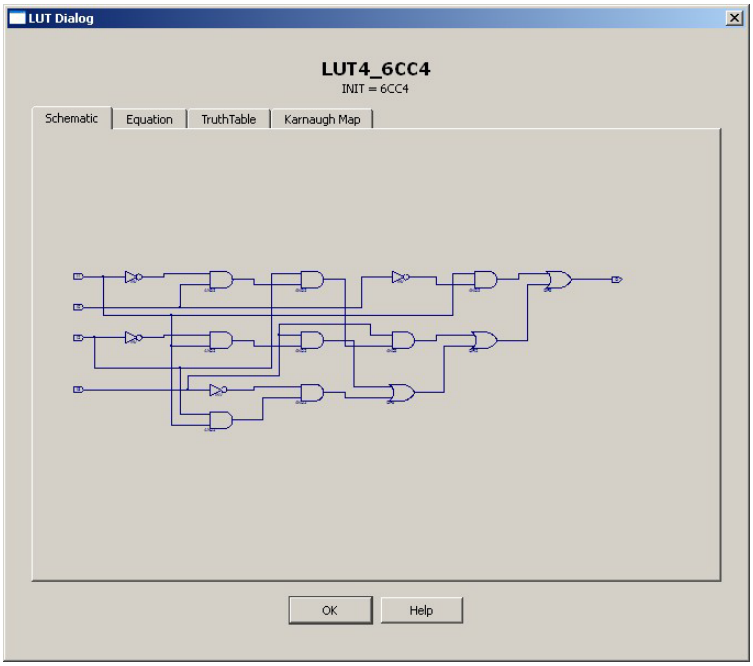


Slika 3.3.5.6. – Kompletan shematski prikaz interne strukture dizajna



Slika 3.3.5.7. – Kompletan shematski prikaz interne strukture dizajna sa stanovišta strukture FPGA čipa

View RTL Schematic podopcija daje prikaz dizajna ne uzimajući u obzir strukturu FPGA čipa, odnosno njegove elemente. Ali, **View Technology Schematic** podopcija u okviru **Synthesize** opcije toka projekta daje prikaz sheme dizajna uzimajući u obzir i strukturu FPGA čipa. Napomenimo da i dalje nije uzet u obzir konačan raspored dizajna po lokacijama FPGA čipa jer nije obavljen proces postavljanja i rutiranja. **View Technology Schematic** podopcija se aktivira na isti način kao i **View RTL Schematic** podopcija, pri čemu se i ovde otvara prozor prikazan na slici 3.3.5.4. Otuda, i ovde možemo da izaberemo kompletan prikaz dizajna ili samo delova dizajna koje smo prethodno selektovali u navigatoru (koji je istog principa kao kod **View RTL Schematic** podopcije). Na slici 3.3.5.7 je prikazan kompletan dizajn sa stanovišta strukture FPGA čipa. I ovde su u donjem delu prozora izlistani delovi dizajna čijom se selekcijom, obeležavaju elementi u shemi koji odgovaraju selektovanom delu. Takođe, osnovne informacije selektovanog dela su prikazane u desnoj polovini donjeg dela prozora. Dvostrukim klikom na kombinacionu logiku (LUT) otvara se prozor sa prikazom kombinacione funkcije koju predstavlja dotični LUT (slika 3.3.5.8). Ovaj prozor sadrži četiri taba. Tab **Schematic** prikazuje shematski izgled realizovane kombinacione funkcije koju predstavlja selektovani LUT. Tab **Equation** prikazuje jednačinu kombinacione funkcije. Tab **Truth Table** prikazuje tablicu istinitosti kombinacione funkcije. Tab **Karnaugh Map** prikazuje izgled Karnoove karte kombinacione funkcije (ova prikaz je moguć samo za LUT-ove do četiri ulaza, za LUT-ove sa većim brojem ulaza ovaj prikaz nije aktivan).



Slika 3.3.5.8. – Izgled kombinacione funkcije selektovanog LUT-a

3.3.6. Funkcionalna simulacija

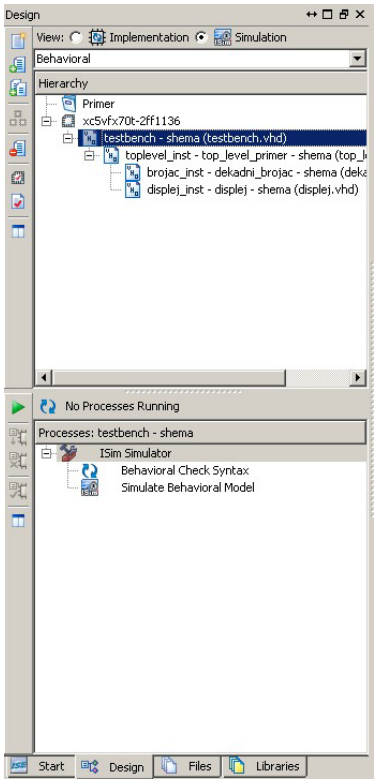
Nakon analize i sinteze poželjno je uraditi funkcionalnu simulaciju dizajna da bi se uverili da dizajn radi korektno sa logičkog stanovišta u idealnim uslovima. ISE softverski paket koristi ugrađeni ISim simulator koji svojim izgledom u velikoj meri podseća na veoma poznati i popularni ModelSim simulator. Da bi pripremili simulaciju neophodno je kreirati odgovarajući testbenč. Pod testbenčom se podrazumeva kreiranje jednog ili više simulacionih VHDL entiteta (u opštem slučaju mogu biti i uključeni entiteti pisani na drugi način, ali u okviru skripti se

pokriva samo VHDL jezik). Top level testbenč fajl je specifičan entitet jer ne sadrži nijedan port i on predstavlja top level entitet za simulaciju (**Simulation View** u strukturi projekta). On samo predstavlja top level omot test okruženja u okviru koga će biti instanciran čitav dizajn u vidu top level entiteta našeg dizajna ili samo deo dizajna (samo pojedini entiteti) koje želimo da testiramo. Unutar top level testbenča treba generisati stimuluse za testirani dizajn. Stimulusi mogu biti generisani u arhitekturi top level simulacionog entiteta ili se mogu kreirati (jedan ili više) entiteti koji će predstavljati generatore stimulus sadržaja za testirani dizajn. Naravno, može se koristiti i kombinacija ova dva pristupa. Kreirajmo testbenč fajl (**testbench.vhd**) kojim ćemo testirati čitav dizajn:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;--neophodno jer radimo sa STD_LOGIC i STD_LOGIC_VECTOR
--top level entitet koji uvozi u vidu komponenti dekadni brojac i displej logiku
ENTITY testbench IS
END testbench;
ARCHITECTURE shema OF testbench IS

	--deklaracija komponente  top level entiteta dizajna
	COMPONENT top_level_primer
	PORT
	(
		clk:		IN STD_LOGIC;--signal takta
		reset:		IN STD_LOGIC;--asinhroni signal reseta
		inc:		IN STD_LOGIC;--kontrolni signal za dozvolu brojanja
		displej_out:	OUT STD_LOGIC_VECTOR(6 DOWNTO 0)--izlaz koji kontrolise displej
	);
	END COMPONENT;
	--signali koje moramo da generisemo da bi testirali dizajn
	SIGNAL clk:STD_LOGIC;
	SIGNAL reset:STD_LOGIC;
	SIGNAL inc:STD_LOGIC;
	SIGNAL displej_out:STD_LOGIC_VECTOR(6 DOWNTO 0);
BEGIN
	--generisanje stimulusa takta
	PROCESS
	BEGIN
		--generisanje takta od 100MHz (perioda 10ns) - 5ns '1', i 5ns '0'
		loop
			clk<='0';
			wait for 5ns;
			clk<='1';
			wait for 5ns;
		end loop;
	END PROCESS;
	--generisanje stimulusa reseta
	PROCESS
	BEGIN
			reset<='1';
			wait for 15ns;--aktivna vrednost reseta traje 15ns, posle se obori na '0'
			loop--neophodno da bi zadrzali konstantno reset na '0'
				reset<='0';
				wait for 5ns;
			end loop;
	END PROCESS;
	--generisanje stimulusa inkrementa
	PROCESS
	BEGIN
```

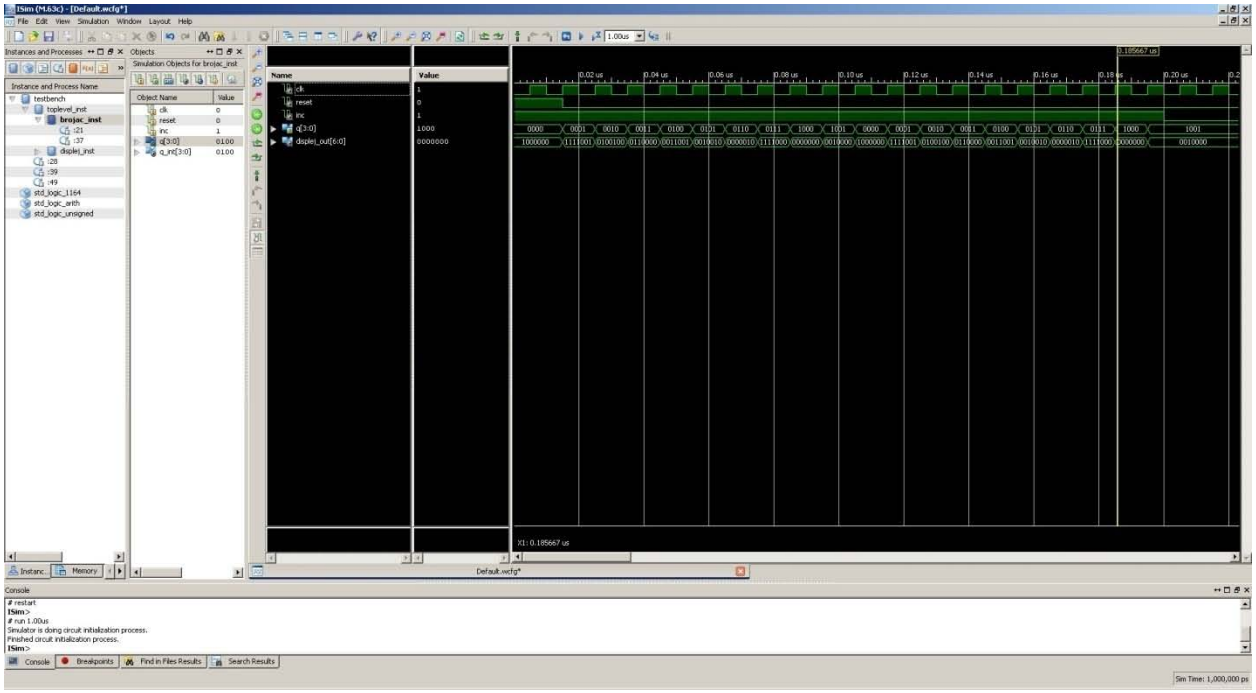
```
                inc<='1';
                wait for 200ns;
                inc<='0';--pauza inkrementa u trajanju od 50ns
                wait for 50ns;
                loop--neophodno da bi zadrzali konstantno inkrement na '1'
                        inc<='1';
                        wait for 5ns;
                end loop;
        END PROCESS;
        --instanciranje komponente top level entiteta dizajna
        toplevel_inst: top_level_primer
        PORT MAP
        (
                clk => clk,
                reset => reset,
                inc => inc,
                displej_out => displej_out
        );
END shema;
```

Kao što vidimo, u testbenču koristimo WAIT FOR konstrukciju za generisanje trajanja signala stimulusa, pri čemu smo takt generisali u vidu petlje jer je on periodičan signal. Petlju smo koristili i za reset i inkrement, ali u njihovom slučaju da bi zadržali konstantnu vrednost tih signala do kraja simulacije. Kada uključimo testbenč fajl u projekat i u strukturi projekta izaberemo **Simulation View** (u padajućem meniju ostavljamo difolt opciju **Behavioral**), prikazaće se hijerarhija simulacionog dizajna gde je testbenč top level entitet za simulaciju (slika 3.3.6.1). U toku projekta će se prikazati meni za simulaciju koji se sastoji od dve komande. Proveru sintakse koda (**Behavioral Check Syntax**), i aktivaciju ISim simulatora (**Simulate Behavioral Model**).



Slika 3.3.6.1. – Simulation View nakon kreiranja i dodavanja testbenč fajla

Klikom na **Behavioral Check Syntax** biće izvršena provera sintakse koda čitavog testbenča (i testbenč fajla i svih fajlova dizajna uključenih u simulaciju). U slučaju otkrivenih grešaka treba ih ispraviti jer nije moguće simulirati dizajn koji sadrži sintaksne greške. Napomenimo da nije potrebno uraditi proces analize i sinteze dizajna pre funkcionalne simulacije dizajna u slučaju ISE okruženja, odnosno ISim simulatora.



Slika 3.3.6.2. – ISim simulator

Klikom na **Simulate Behavioral Model** otvara se prozor ISim simulatora (slika 3.3.6.2). Sa leve strane prozora postoje dve kolone. Prva kolona (**Instance and Process Name**) daje mogućnost selekcije neke instance simulacionog dizajna, pri čemu su instance složene hijerarhijski i na vrhu se nalazi testbenč entitet, odnosno njegova instanca. Iznad ove kolone se nalaze ikonice koje omogućuju filtriranje prikaza ove kolone, na primer možemo selektovati prikaz samo VHDL entiteta. Po difoltu su uključene sve opcije (ikonice). Druga kolona (**Simulation Objects**) daje pregled portova i internih signala selektovane instance entiteta u prvoj koloni. Iznad ove kolone se takođe nalaze ikonice za filtriranje prikaza ove kolone. Na primer može se selektovati prikaz samo ulaznih portova selektovanog entiteta. I ovde su po difoltu uključene sve opcije (ikonice). Sam prozor koji prikazuje rezultat simulacije (**Wave Window**) se sastoji iz tri celine. Prva kolona daje listu posmatranih signala. Druga kolona daje vrednost posmatranih signala u selektovanoj tački (gde je postavljen kurzor - žuta linija u trećoj celini). Treća celina daje grafički prikaz ponašanja signala tokom simulacije. *Drag & drop* opcijom možemo prevlačiti signale koje želimo posmatrati iz kolone **Simulation Objects** u prvu celinu **Wave Window** prozora (gde se nalazi lista prikazanih signala). Drugi način je desni klik na željeni signal koji želimo posmatrati i izbor opcije **Add to Wave Window**. Sam redosled signala u listi posmatranih signala je moguće promeniti pomoću miša jednostavnim vučenjem signala na željeno mesto. U donjem delu prozora se ispisuju poruke generisane tokom simulacije. Ukoliko naiđe na neku grešku, simulator će stati sa svojim radom, a poruka o grešci će biti ispisana u donjem delu prozora. Takođe, ako je izabran tab **Console** moguće je tekstualno unositi komande za definisanje rada ISim simulatora (na primer restartovanje simulacije), ali većina

komandi je dostupna preko glavnog menija i ikonica. U sekciji koja bude obrađivala analizu potrošnje snage, biće prikazane komande za generisanje fajla koji sadrži aktivnosti signala i koje se tekstualno unose.

U koloni ikonica sa leve strane **Wave Window** prozora se nalaze opcije za brzo aktiviranje često korišćenih opcija poput zumiranja prikaza, pomeranja kursora na početak/kraj simulacije, pomeranja kursora na prethodnu ili sledeću tranziciju (za ovu opciju mora biti selektovan signal u listi posmatranih signala) i sl. Većina ovih opcija se nalazi i u redu ikonica ispod glavnog menija. Format prikaza vrednosti signala se bira desnim klikom na željeni posmatrani signal i izborom opcije **Radix** čime se otvara podmeni sa mogućim formatima prikaza signala (binarni, heksadecimalni, ASCII i dr.).

U okviru **Simulation** grupe opcija u glavnom meniju su nalaze komande za pokretanje simulacije poput pokretanja simulacije ili restartovanja simulacije. Ove komande su izvučene i kao ikonice u redu ikonica ispod glavnog menija (slika 3.3.6.3). Prva po redu ikonica radi restart simulacije. Druga ikonica pokreće simulaciju. Treća ikonica u kombinaciji sa edit poljem pokreće simulaciju koja traje onoliko vremena koliko je specificirano u edit polju. Sledeća ikonica označava jedan korak simulacije, odnosno ona traje koliko i najkraće definisano ponašanje u simulaciji (u našem primeru 5ns). Poslednja ikonica se koristi za nasilno zaustavljanje (pauziranje) simulacije.

Slika 3.3.6.3. – Ikonice za poketanje simulacije

Na kraju napomenimo da se upotrebom opcije **Generate Post-Synthesis Simulation Model** iz toka projekta (pomenutoj u sekciji 3.3.4) generiše simulacioni modul dizajna koji u obzir uzima rezultat procesa analize i sinteze. Ovako generisani modul se može uključiti u testbenč i na taj način testirati dizajn koji je rezultat procesa analize i sinteze originalnog dizajna. Ovaj modul (njegov VHDL fajl) se kreira i smešta u folder */netgen/synthesis*. Naziv entiteta ovog modula se podešava u podešavanjima opcije **Generate Post-Synthesis Simulation Model** koji se može otvoriti desnim klikom na ovu opciju i potom izborom **Process Properties** u meniju koji se otvara desnim klikom. Difolt naziv modula je identičan nazivu top level entiteta dizajna, što pravi problem prilikom uključivanja generisanog modula u funkcionalnu simulaciju pa je preporuka modifikovati naziv modula dodavanjem jednog karaktera ili cifre na difolt naziv, a potom i iste takve modifikacije naziva komponente koja odgovara top level entitetu dizajna u testbenč fajlu. Takođe je potrebno promeniti i naziv VHDL fajla u kome se nalazi generisani simulacioni modul i potom taj fajl uključiti u projekat. Tek potom se može aktivirati simulacija u ISim simulatoru. U principu upotreba ovog simulacionog modula nije potrebna, jer funkcionalna simulacija sa originalnim dizajnom je sasvim adekvatna za gotovo sve slučajeve.

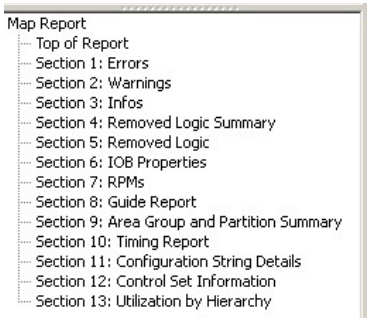
3.3.7. Proces postavljanja i rutiranja i proces vremenske analize

U ISE softverskom paketu, proces postavljanja i rutiranja se sastoji od tri koraka: prevođenja (**Translate**), mapiranja (**Map**) i postavljanja i rutiranja (**Place & Route**). Ulogu svakog od koraka smo objasnili u sekciji 3.3.4. Sva tri koraka se mogu pokrenuti odjednom (koraci će se izvršavati jedan za drugim, redosledom navedenim u prethodnoj rečenici) tako što se uradi dvostruki klik na opciju **Implement Design** u toku projekta ili selekcijom opcije **Implement Design** i potom izbora opcije **Process->Run** iz glavnog menija. Na identičan način se može pokrenuti samo jedan od koraka, ali tada se umesto **Implement Design** opcije bira se

odgovarajuća podopcija, odnosno željeni korak procesa postavljanja i rutiranja u toku projekta. Naravno, ako neki od prethodnih koraka kompajliranja nisu urađeni prvo će da budu urađeni prethodni koraci kompajliranja pa tek onda korak koji smo aktivirali.

Kada se izvrši proces prevođenja dodaje se izveštaj **Translation Report** u listu izveštaja prikazanoj na slici 3.3.5.2. Ovaj izveštaj je veoma jednostavan i daje podatke o ulaznom fajlu (koji je generisan kao rezultat procesa analize i sinteze), kao i o korisničkim ograničenjima uključenim u projekat u vidu .ucf fajlova, na osnovu kojih se generiše fajl sa ekstenzijom .ngd koji se koristi u sledećem koraku procesa postavljanja i rutiranja. U izveštaju se navodi naziv kreiranog fajla, kao i koliko je proces prevođenja trajao.

U okviru koraka prevođenja postoji i dodatna opcija za generisanje simulacionog modula dizajna nakon izvršenog koraka prevođenja (**Generate Post-Translation Simulation Model**). Ova opcija se pokreće na identičan način kao i korak prevođenja, uz razliku da se selektuje ova opcija. Kreirani modul (njegov VHDL fajl) se kreira i smešta u folder */netgen/translate*. U ovom slučaju je najbolje ostaviti difolt podešavanje naziva generisanog simulacionog modula da bude identično nazivu top level entita dizajna. Simulacija ovog simulacionog modula se aktivira izborom opcije **Post-Translate** padajućeg menija u okviru **Simulation View**, pri čemu ako se poklapaju nazivi generisanog simulacionog modula i originalnog top level entiteta dizajna, ne moraju se vršiti nikakve izmene u testbenču (ne treba raditi izmenu naziva komponente koja odgovara top level entitetu dizajna). Napomenimo još da nema preteranog smisla koristiti ovaj modul jer posle koraka prevođenja dizajn još nije razmešten po FPGA čipu.



Slika 3.3.7.1. – Struktura detaljnog izveštaja koraka mapiranja

Nakon aktiviranja i izvršavanja koraka mapiranja dodaje se izveštaj **Map Report** u listu izveštaja prikazanoj na slici 3.3.5.2. Struktura ovog izveštaja je data na slici 3.3.7.1. Na vrhu izveštaja (**Top of Report**) se daje sažeti rezultat rada koraka mapiranja gde se navodi FPGA čip na koji je mapiran dizajn, a potom i zbirni pregled iskorišćenih resursa FPGA čipa, kao i vreme trajanja koraka mapiranja. Izveštaji **Errors** i **Warnings** daju pregled grešaka, odnosno upozorenja koja su se javila u toku koraka mapiranja. Naravno, ako se javila greška, proces mapiranja će biti prekinut, pa ispitivanjem greške treba korigovati dizajn na odgovarajući način da bi se izbegla dotična greška. Izveštaj **Infos** daje pregled obaveštenja koja su se javila u koraku mapiranja poput na primer deklarisanih radnih uslova okoline. Izveštaj **Removed Logic** daje pregled uklonjene logike iz mapiranog dizajna, dok **Removed Logic Summary** daje zbirni izveštaj uklonjene logike. Ako ovi izveštaji nisu prazni tj. ako je neka logika uklonjena treba proveriti da li je zaista dotična logika trebala biti uklonjena ili ne. Ako nije trebala biti uklonjena, onda je u pitanju greška u dizajnu koju treba otkloniti. **IOB Properties** izveštaj daje pregled konfiguracije ulaznih/izlaznih blokova, koji se nalaze uz pinove FPGA čipa (preko njih se vezuju portovi top level dizajna na pinove FPGA čipa), dodeljenih portovima top level entiteta. Izveštaj

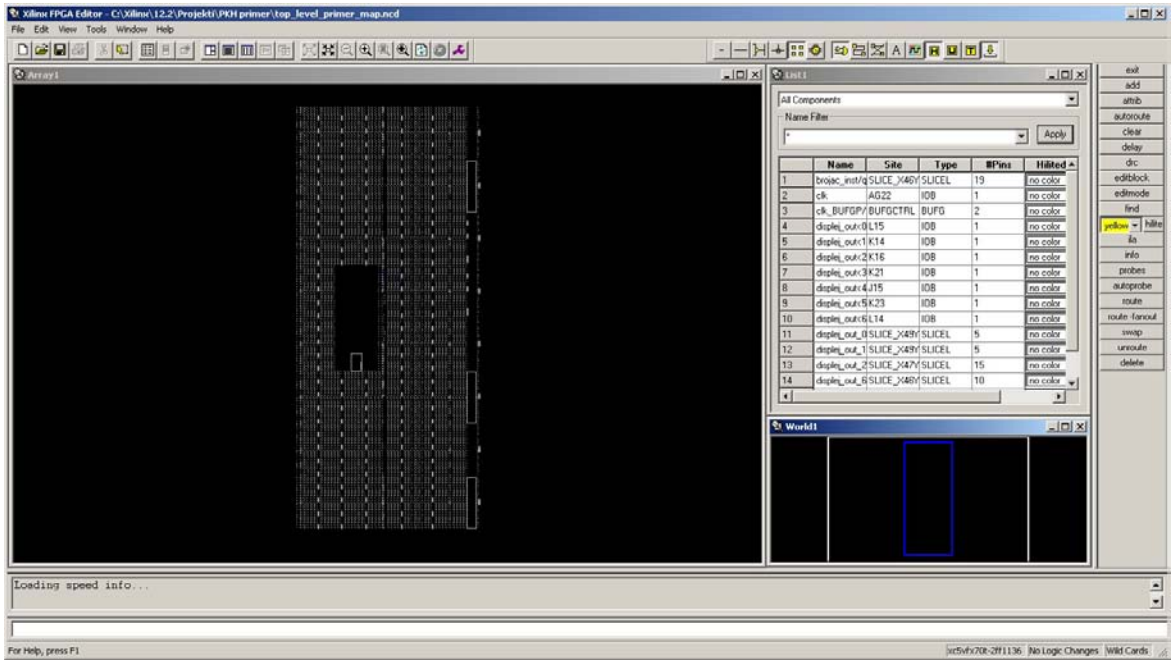
daje pregled njihovih konfigurisanih karakteristika poput, na primer, konfigurisanog smera (ulazni, izlazni ili bidirekcioni). **RPMs** izveštaj se odnosi na tzv. relativne makroe (*Relationally Placed Macros* - RPMs) u dizajnu. RPMs se koriste za slučaj kada za deo logike želimo da zadržimo njihove međusobne pozicije (razmeštaj po FPGA čipu), a onda tako formirani deo logike razmeštamo po čipu, pri čemu njihove međusobne pozicije ostaju iste - otuda pojam relativni. Ovi makroi su zgodni kada želimo posle ručno da modifikujemo postavljeni dizajn na FPGA čipu radi ostvarivanja boljih performansi u pogledu kašnjenja u dizajnu i maksimalne frekvencije dizajna. **Area Group and Partition Summary** daje sažeti pregled vremenskih oblasti i particija u dizajnu. Pod vremenskom oblasti se podrazumeva oblast koju kontroliše grupa taktova. Svaki FPGA čip sadrži brze takt linije i takođe svaki FPGA čip ima definisan maksimalni broj taktova koje može da podrži. Međutim, kod nekih FPGA čipova se ne razvlače sve brze takt linije do svih delova FPGA čipa, nego manji deo linija. To znači da do pojedinih delova FPGA čipa ne možemo dovesti sve moguće taktove, već samo deo njih. U slučaju da u dizajnu koristimo velik broj taktova to znači da u pojedine delove FPGA čipa potencijalno ne možemo proizvoljno da rasporedimo delove dizajna jer se može desiti da ne možemo dovući do njih sve signale takta. Tada je zgodno definisati tzv. vremenske oblasti u FPGA čipu na koje logički razdvajamo delove dizajna. U svaku vremensku oblast dolazi deo taktova i u njih se razmešta samo ona logika na koju utiču ti taktovi (preciznije koji rade po diktatu tih taktova). Particije dizajna i njihova uloga je već objašnjena u okviru sekcije 3.2.17. **Timing Report** daje informaciju kako generisati vremensku analizu dizajna nakon koraka mapiranja, pri čemu se ukazuje na to da se radi o proceni kašnjenja u dizajnu, jer još nije završen kompletan proces postavljanja i rutiranja, odnosno još nije dizajn raspoređen na konkretne lokacije FPGA čipa. **Control Set Information** daje listu kombinacija kontrolnih signala za registre (preciznije flip-flopove). U pitanju je kombinacija signala takta, signala reseta, signala seta i signala dozvole za flip-flop. Izveštaj navodi sve kombinacije detektovane u dizajnu i za svaku kombinaciju navodi broj flip-flopova na koje utiče, kao i broj slajsova na koje utiče (setimo se iz prvog poglavlja skripti da jedan slajs sadrži više flip-flopova). **Utilization by Hierarchy** izveštaj daje pregled iskorišćenih resursa hijerarhijski po delovima (entitetima) dizajna. Da bi se generisali izveštaji **Control Set Information** i **Utilization by Hierarchy** neophodno je aktivirati opciju **Generate Detailed MAP Report** u podešavanjima koraka mapiranja.

Pokretanje vremenske analize dizajna nakon koraka mapiranja se vrši dvostrukim klikom na podopciju **Generate Post-Map Static Timing** (ili selekcijom ove opcije i izborom **Process-> Run** iz glavnog menija). Izveštaj ove vremenske analize je stavljen među sekundarne izveštaje (**Secondary Reports**) pod nazivom **Post-Map Static Timing Report**. Međutim, pošto je u pitanju samo procena jer dizajn još nije razmešten na konkretne lokacije FPGA čipa, ova vremenska analiza nema preteranu upotrebnu vrednost i bolje je uraditi vremensku analizu nakon obavljenog koraka postavljanja i ruitranja. U principu se vremenska analiza posle koraka mapiranja preporučuje za detekciju kritičnih putanja kod kojih je dominantno kašnjenje usled kašnjenja kroz kombinacionu logiku, pa možemo izvršiti, na primer, modifikaciju dizajna radi smanjenja nivoa kombinacione logike na kritičnoj putanji.

U okviru koraka mapiranja postoji i dodatna opcija za generisanje simulacionog modula dizajna nakon izvršenog koraka mapiranja (**Generate Post-Map Simulation Model**). Ova opcija se pokreće na identičan način kao i korak mapiranja, uz razliku da se selektuje ova opcija. Kreirani modul (njegov VHDL fajl, kao i .sdf fajl) se kreira i smešta u folder */netgen/map*. Fajl sa ekstenzijom .sdf sadrži procene kašnjenja u dizajnu. Kao i u slučaju simulacionog modula generisanog nakon koraka prevođenja, i ovde je najbolje ostaviti difolt podešavanje naziva

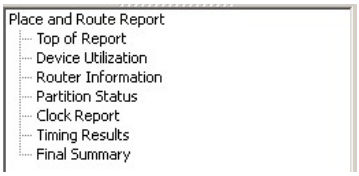
generisanog simulacionog modula iz identičnih razloga. Simulacija ovog simulacionog modula se aktivira izborom opcije **Post-Map** padajućeg menija u okviru **Simulation View**. Napomenimo još da nema preteranog smisla koristiti ovaj simulacioni modul jer posle koraka mapiranja dizajn još nije razmešten po konkretnim lokacijama FPGA čipa tako da se ovaj modul ne može koristiti kao deo precizne vremenske simulacije.

Podopcija **Manually Place & Route** aktivira FPGA editor koji daje prikaz FPGA čipa i rasporeda dizajna na njemu (slika 3.3.7.2). U FPGA editoru možemo izvršiti ručno postavljanje i raspoređivanje dizajna na sam FPGA čip. FPGA editor možemo koristiti, na primer, ukoliko je narušeno vremensko ograničenje, a vizuelno smo utvrdili način da rasporedimo dizajn tako da sprečimo narušavanje vremenskog ograničenja. U okviru ovih skripti se neće obrađivati FPGA editor i način rada u tom alatu.



Slika 3.3.7.2. – FPGA editor

Nakon aktiviranja i izvršavanja koraka postavljanja i rutiranja dodaje se izveštaj **Place and Route Report**, kao i izveštaj **Post-PAR Static Timing Report** u listu izveštaja prikazanoj na slici 3.3.5.2. Struktura izveštaja **Place and Route Report** je data na slici 3.3.7.3.



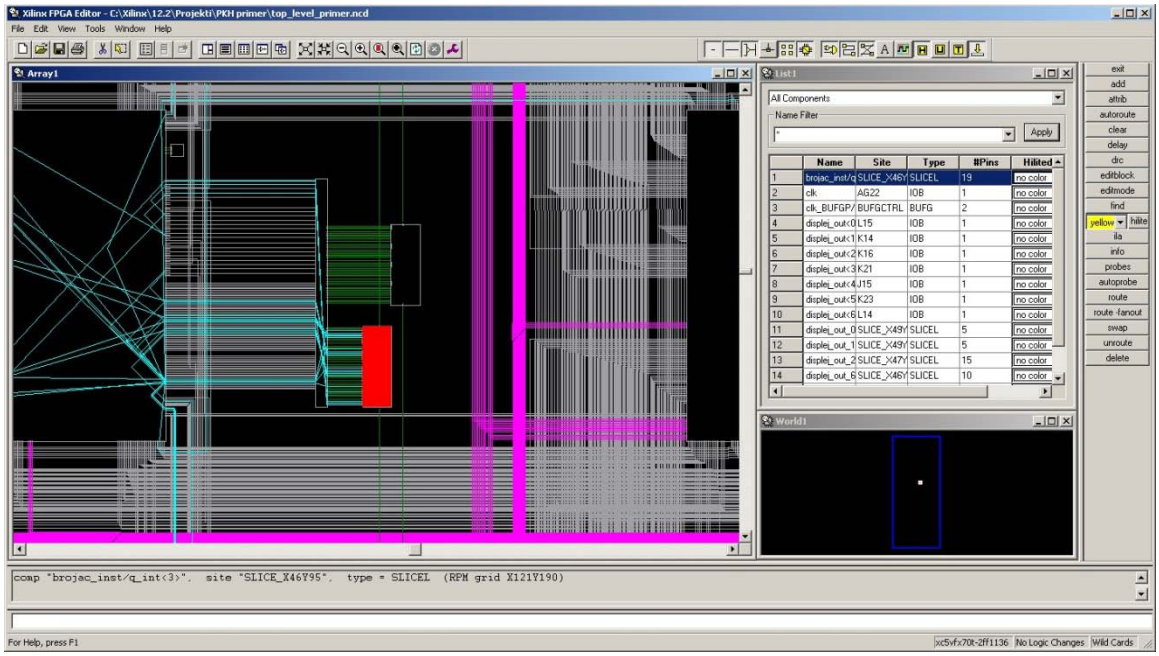
Slika 3.3.7.3. – Struktura detaljnog izveštaja koraka postavljanja i rutiranja

Top of Report izveštaj daje pregled osnovnih podešavanja, poput FPGA čipa i radnih uslova. **Device Utilization** daje zbirni pregled iskorišćenih resursa FPGA čipa. **Router Information** daje pregled rada procesa rutiranja kroz etape ovog procesa. U početnim etapama deo putanja još nije postavljen, dok u naknadnim etapama sve putanje (linije dizajna) su izrutirane i u tim etapama se traži efikasnije rešenje (nalaženje bolje rute za neku putanju). **Partition Status** daje informacije o particijama u dizajnu. **Clock Report** daje informacije o

taktovima u dizajnu. **Timing Results** daje pregled vremenskih ograničenja za taktove (ako korisnik nije postavio vremenska ograničenja, onda kompajler automatski generiše sam vremenska ograničenja). Simbol * kod vremenskog ograničenja znači da je dotično vremensko ograničenje narušeno. **Final Summary** sadrži finalne napomene u vidu broja poruka grešaka, poruka upozorenja i informacionih poruka generisanih tokom koraka postavljanja i rutiranja. Takođe, sadrži trajanje koraka postavljanja i rutiranja, kao i naziv rezultujućeg fajla generisanog nakon koraka postavljanja i rutiranja.

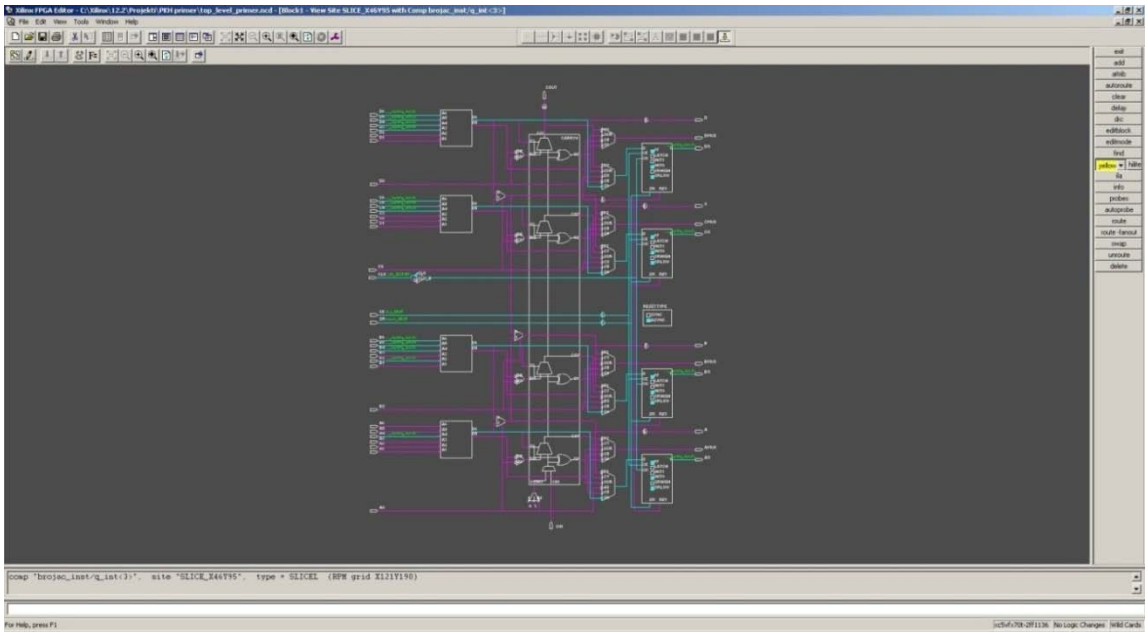
Post-PAR Static Timing Report izveštaj sadrži vremensku analizu dizajna. Između ostalog sadrži listu najkritičnijih putanja u dizajnu, pri čemu su one grupisane u logičke grupe. **Clock to Pad** označava vreme koje protekne od ivice takta na koju se menja sadržaj flip-flopa povezanog na izlazni pin pa do pojave nove vrednosti na tom pinu. Pri tome izlaz flip-flopa može biti direktno vezan na izlazni pin ili preko kombinacione logike. **Clock to Setup** označava vreme koje protekne od ivice takta na koju se menja sadržaj flip-flopa povezanog na drugi flip-flop pa do pojave nove vrednosti na ulazu tog drugog flip-flopa plus vreme postavljanja signala na tom drugom flip-flopu. Flip-flopovi mogu biti direktno vezani ili preko kombinacione logike. **Setup/Hold** sadrži maksimalno vreme postavljanja i držanja signala koji reaguju na dotični signal takta. Podsetimo se, vreme postavljanja je vreme za koje signal mora biti prisutan na ulazu flip-flopa pre nailaska aktivne ivice takta, a vreme držanja je vreme za koje signal mora zadržati svoju vrednost nakon aktivne ivice takta. Ova dva vremenska ograničenja moraju biti zadovoljena da bi flip-flop pouzdano menjao svoj sadržaj, odnosno reagovao na aktivnu ivicu takta.

Podopcija **Generate Post-Place & Route Static Timing** koraka postavljanja i rutiranja se automatski aktivira sa aktivacijom koraka postavljanja i rutiranja i predstavlja preciznu vremensku analizu dizajna. Kao rezultat se generiše već pomenuti **Post-PAR Static Timing Report** izveštaj.

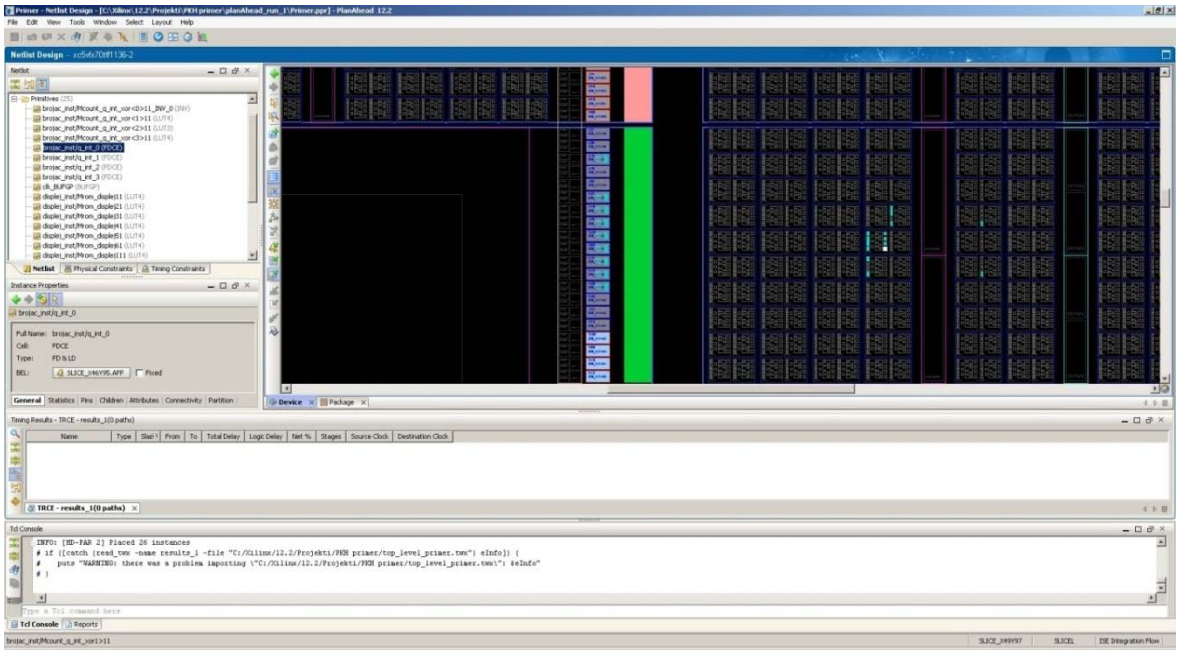


Slika 3.3.7.4. – Prikaz dekadnog brojača iz dizajna primera u FPGA editoru

Podopcija **View/Edit Routed Design** otvara već pominjani FPGA editor sa prikazom trenutnog razmeštaja dizajna na FPGA čipu (slika 3.3.7.4). Razmeštaj je rezultat koraka postavljanja i rutiranja. Pomoću FPGA editora možemo vizuelno videti raspored dizajna na FPGA čipu, ali možemo i modifikovati ovaj raspored ako to želimo. Na slici 3.3.7.4 je dat zumiran prikaz, gde crveni blok (slajs) označava lokaciju brojača u dizajnu. Dvostrukim klikom na neki blok (slajs) se dobija interna struktura tog bloka, kao i njegova konfiguracija. Slika 3.3.7.5 prikazuje internu strukturu i konfiguraciju slajsa koji sadrži dekadni brojač.



Slika 3.3.7.5. – Prikaz interne strukture slajsa koji sadrži dekadni brojač



Slika 3.3.7.6. – PlanAhead alat

Xpower Analyzer podopcija predstavlja alat za procenu potrošnje snage dizajna. O ovom alatu će biti reči u kasnijim sekcijama. **Analyze Timing/Floorplan Design** podopcija

pokreće alat **PlanAhead** koji takođe daje uvid u raspored dizajna na FPGA čipu (slika 3.3.7.6). Ovaj alat takođe omogućava modifikaciju rasporeda dizajna na FPGA čipu. Može se reći da **PlanAhead** predstavlja moćniju alternativu FPGA editoru. Zbog kompleksnosti, ali i brojnih mogućnosti ovog alata, **PlanAhead** neće biti razmatran u okviru ovih skripti.

U okviru koraka postavljanja i rutiranja se takođe nalazi podopcija za generisanje simulacionog modula dizajna nakon izvršenog koraka postavljanja i rutiranja (**Generate Post-Place & Route Simulation Model**). Ova podopcija se pokreće na identičan način kao i ostale opcije i podopcije koje smo prethodno opisivali. Kreirani simulacioni modul (njegov VHDL fajl, kao i .sdf fajl) se kreira i smešta u folder */netgen/par*. Fajl sa ekstenzijom .sdf sadrži kašnjenja u dizajnu koja su dobijena vremenskom analizom nakon okončanja koraka postavljanja i rutiranja. I u ovom slučaju generisanja simulacionog modula je najbolje ostaviti difolt podešavanje naziva generisanog simulacionog modula iz identičnih razloga kao i u prethodna dva slučaja (simulacioni moduli za korake prevođenja i mapiranja). Simulacija ovog simulacionog modula se aktivira izborom opcije **Post-Route** padajućeg menija u okviru **Simulation View**. Ovaj generisani simulacioni modul ćemo koristiti u sekciji koja opisuje proces vremenske simulacije.

Podopcija **Generate IBIS Model** kreira simulacioni model dizajna po IBIS (*Input Output Buffer Information Specification*) specifikaciji. IBIS modeli se koriste u simulacijama veza između čipova i ponašanja signala na tim interkonekcijama između čipova. Kao rezultat ove podopcije se kreira fajl sa ekstenzijom .ibs u korenu radnog direktorijuma projekta koji predstavlja IBIS model dizajna, odnosno, preciznije FPGA čipa. Ova podopcija u sebi sadrži **View IBIS Model** mogućnost kojom se otvara sadržaj kreiranog fajla u ISE tekstualnom editoru.

Podopcija **Back-annotate Pin Locations** kreira .ucf fajl koji sadrži lokacije pinova za koje su vezani portovi top level dizajna po rasporedu koji je napravio proces postavljanja i rutiranja. Naravno, ovo se odnosi samo na one portove top level dizajna za koje korisnik nije prethodno izvršio dodelu pinova. Kreirani .ucf fajl se smešta u koren radnog direktorijuma i istovremeno se dodaje u projekat. Međutim, ova podopcija zahteva da se ponovo aktivira proces postavljanja i rutiranja (opcija **Implement Design**) koja će u novom prolazu uzeti u obzir kreirani .ucf fajl kao korisnički definisano ograničenje. U okviru ove podopcije postoji i **View Locked Pin Constraints** čijim pokretanjem se daje samo prikaz rasporeda pinova dizajna (preciznije pinova na koje su vezani portovi top level dizajna) u ISE tekstualnom editoru, pri čemu se ne pokreće podopcija **Back-annotate Pin Locations** i samim tim se ne kreira .ucf fajl. Ovako kreiran prikaz rasporeda pinova se čuva u fajlu sa ekstenzijom .lpc. Pošto u ovom slučaju nije kreiran .ucf fajl, ne mora ponovo da se pokreće proces postavljanja i rutiranja.

Podopcija **Generate Power Data**, kao i **Xpower Analyzer** alat daje procenu potrošnje dizajna i biće objašnjena u narednim sekcijama.

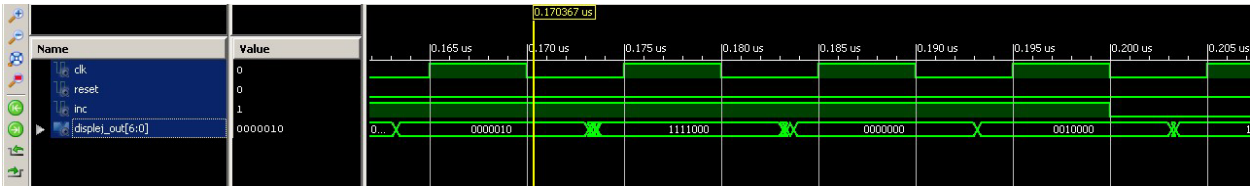
3.3.8. Proces generisanja konfiguracionog fajla FPGA čipa

Nakon procesa postavljanja i rutiranja, treba pokrenuti proces generisanja konfiguracionog fajla FPGA čipa. Ovaj proces se pokreće dvostrukim klikom na opciju **Generate Programming File** u toku projekta ili selektovanjem te opcije i potom aktiviranjem **Process->Run** iz glavnog menija. Kao rezultat procesa se dobija fajl sa ekstenzijom .bit koji se može koristiti za konfigurisanje FPGA čipa tako da obavlja funkciju projektovanog dizajna. Naravno, pored .bit fajla mogu biti generisani i drugi formati konfiguracionog fajla ako smo tako

selektovali u podešavanjima ovog procesa. Takođe, u tim podešavanjima možemo podesiti i da se ne generiše .bit format.

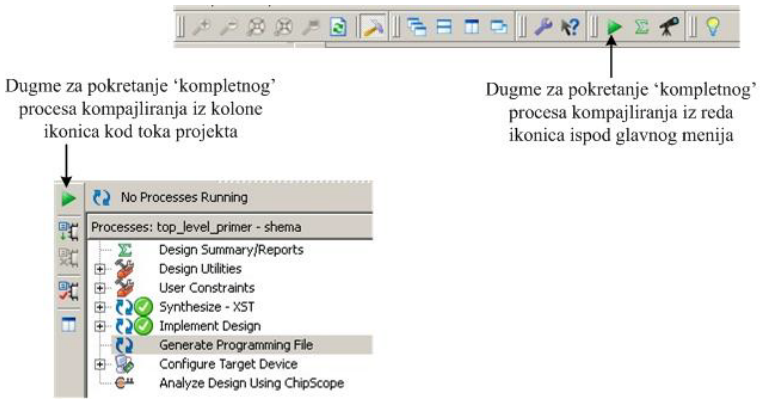
3.3.9. Vremenska simulacija

U okviru vremenske simulacije ćemo koristiti simulacioni modul kreiran primenom opcije **Generate Post-Place & Route Simulation Model**, koja je navedena u sekciji 3.3.7. Da bi se aktivirala vremenska simulacija, biramo **Simulation View** prikaz, a u padajućem meniju biramo opciju **Post-Route**. Ako je naziv entiteta simulacionog modula identičan nazivu top level entiteta (difolt podešavanje), onda ne treba ništa da se menja u testbenč fajlu datom u sekciji 3.3.6. U suprotnom je neophodno izvršiti korekciju naziva komponente koja predstavlja top level entitet dizajna na ime koje je dato generisanom simulacionom modulu. Vremenska simulacija u ISim simulatoru se pokreće na identičan način kako smo pokretali i funkcionalnu simulaciju. Na primer, dvostrukim klikom na opciju **Simulate Post-Place & Route Model**. Na slici 3.3.9.1 je prikazana vremenska simulacija dizajna. Opis ISim simulatora dat u sekciji 3.3.6 važi i u ovom slučaju. Praktično jedina razlika je što su sada uzeta realna kašnjenja linija i logike u dizajnu. Kao što vidimo sa slike 3.3.9.1, u vremenskoj simulaciji se primećuju gličevi koji nisu bili prisutni u funkcionalnoj simulaciji.



Slika 3.3.9.1. – Vremenska simulacija u ISim simulatoru

3.3.10. Pokretanje kompletnog procesa kompajliranja



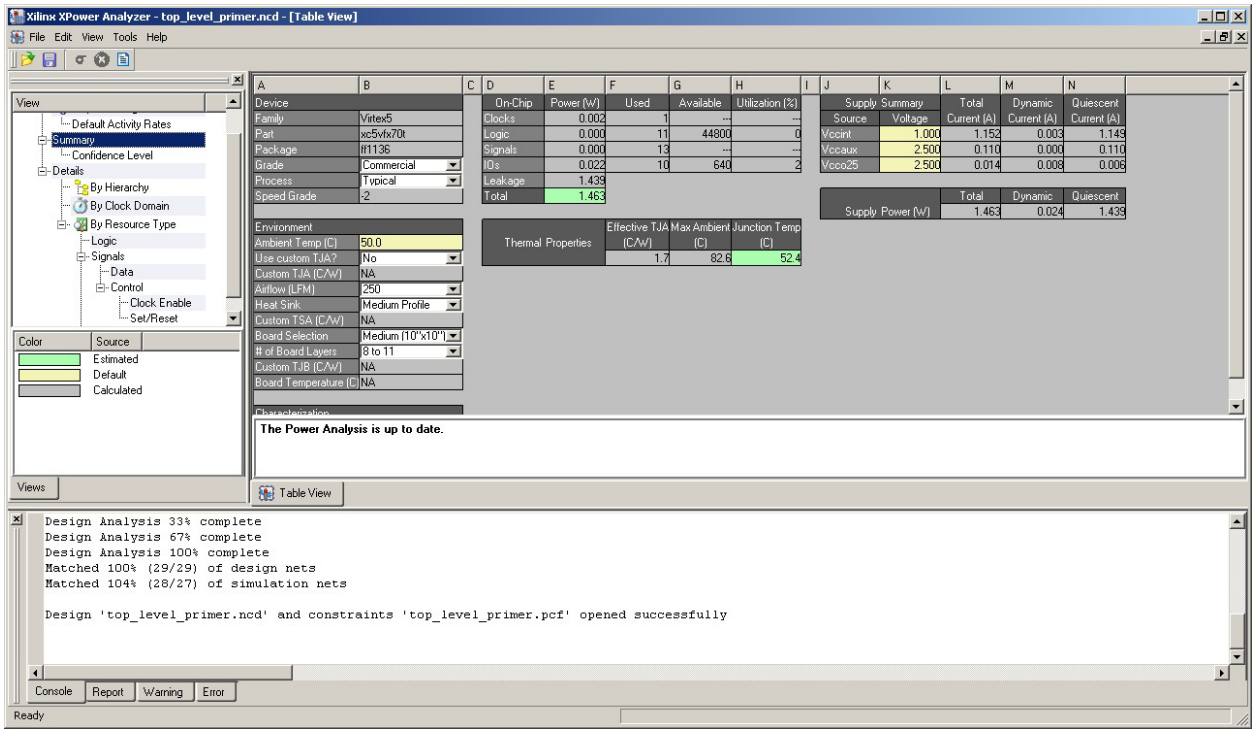
Slika 3.3.10.1. – Pokretanje kompletnog procesa kompajliranja

U prethodnim sekcijama smo prolazili kroz korake procesa kompajliranja i navodili kako se svaki korak ponaosob aktivira. Međutim, često je zgodno odmah pokrenuti kompletan proces kompajliranja. Kompletan proces kompajliranja se pokreće tako što se u strukturi projekta selektuje top level entitet dizajna (podrazumeva se da je izabran **Implementation View** prikaz) pa se tek onda pokrene proces kompajliranja. Proces kompajliranja se pokreće klikom na odgovarajuće dugme iz reda ikonica ispod glavnog menija (slika 3.3.10.1), klikom na odgovarajuću ikonicu u toku projekta (slika 3.3.10.1), ili izborom opcije iz glavnog menija **Process->Implement Top Module**. Pokretanjem kompletnog procesa kompajliranja izvršiće se

proces analize i sinteze, proces postavljanja i rutiranja i proces vremenske analize, ali ne i proces generisanja konfiguracionog fajla FPGA čipa. Da bi se zaista izvršio kompletan proces kompajliranja koji uključuje i generisanje konfiguracionog fajla FPGA čipa, mora se dvostruko kliknuti na opciju **Generate Programming File** u toku projekta ili selektovati navedenu opciju i potom aktivirati opciju **Process->Run** iz glavnog menija. Tada će se izvršiti svi koraci kompajliranja pre generisanja konfiguracionog fajla, a potom i samo generisanje konfiguracionog fajla FPGA čipa.

3.3.11. Analiza potrošnje snage dizajna

Prozor za analizu potrošnje snage dizajna, prikazan na slici 3.3.11.1, otvaramo izborom opcije **Tools->XPower Analyzer** iz glavnog menija. Ovaj alat se može pokrenuti aktivacijom podopcije **XPower Analyzer** koja se nalazi u okviru **Place & Route** koraka u toku projekta. U koloni sa leve strane ovog prozora se nalazi brauzer za izbor prikaza rezultata analize ili različitih podešavanja parametara koji se otvaraju sa desne strane u glavnom delu prozora. Ispod brauzera za izbor prikaza se nalazi legenda za boje koje se koriste u označavanju pojedinih vrednosti u tabelama koje su prikazane u centralnom delu prozora. U donjoj polovini prozora se ispisuju poruke pri čemu postoji više tabova za filtriranje prikaza poruka. Kada se promeni jedan ili više parametara, potrebno je ponovo pokrenuti proces analize izborom opcije **Tools->Update Power Analysis** iz glavnog menija prozora za analizu potrošnje snage.



Slika 3.3.11.1. – Prozor za analizu potrošnje snage

Kada se pokrene **XPower Analyzer** alat, odmah se izvršava analiza snage dizajna i otvara se u glavnom delu prozora zbirni rezultat analize kao što je prikazano na slici 3.3.11.1. **Device** tabela daje pregled izabranog čipa i mogućnost podešavanja čipa poput na primer da li je u pitanju komercijalna ili industrijska verzija čipa i sl. **Enviroment** tabela daje pregled radnih uslova korišćenih u okviru analize, kao i mogućnost njihovih podešavanja. **On-Chip** tabela daje pregled potrošnje snage čipa po delovima dizajna (takt, interna logika i signali, ulazno/izlazni

blokovi, curenje). **Supply Summary** tabela daje pregled izvora napajanja koje koristi čip, njihove napone, kao i procene jačine struja koje generišu ti izvori (ukupna, dinamička i statička). Napomena, pojam *quiescent* se odnosni na statičku komponentu. **Supply Power** tabela daje pregled ukupne potrošnje snage, kao i dinamičke i statičke komponente. **Thermal Properties** daje pregled termičkih karakteristika koje su pretpostavljene u analizi poput maksimalne temperature okoline. Klikom na neka polja u tabelama, otvara se mogućnost unosa nove vrednosti tj. promene trenutne vrednosti parametra (u pitanju su gotovo sva obojena polja). Nova (ručno) uneta vrednost parametra dobija prioritet nad prethodnom vrednošću parametra kako god ona bila dobijena. Napomene iz prethodne dve rečenice važe za sve prikaze koji će u nastavku ove sekcije biti opisani.

Ako u brauzeru sa leve strane izaberemo opciju **View->Project Settings->Default Activity Rates**, u centralnom delu prozora se otvara tabela sa mogućnošću definisanja inteziteta promene signala. Za sve signale koji nisu definisani zasebno, poput zapisane aktivnosti signala koja je uključena u analizu preko nekog fajla, koristiće se ovde podešene vrednosti za aktivnost signala u okviru analize potrošnje snage dizajna. Nova vrednost, za neki parametar koji želimo promeniti, se unosi klikom na polje sa starom vrednošću parametra i potom unosa nove vrednosti.

Opcija **View->Summary** daje već opisani zbirni pregled rezultata analize koji je prikazan na slici 3.3.11.1. Podopcija **View->Summary->Confidence Level** prikazuje nivo poverenja rezultata analize.

Name	Power (W)*	Logic Power (W)	Signal Power (W)	# FFs	# LUTs
Hierarchy total	**0.00056**	0.00016	0.00040	4	11
top_level_primer	**0.00028 / 0.00056**	0.00000 / 0.00016	0.00028 / 0.00040	0 / 4	0 / 11
brojac_inst	**0.00020**	0.00009	0.00012	4	4
displej_inst	**0.00007**	0.00007	0.00000	0	7

Slika 3.3.11.2. – Potrošnja po entitetima dizajna

U okviru opcije **View->Details** nalazi se niz prikaza koji daju detaljniji uvid u rezultate analize potrošnje snage dizajna. Prikaz **By Hierarchy** daje prikaz potrošnje snage po entitetima dizajna, pri čemu se daje i prikaz potrošnje po logici i signalima (slika 3.3.11.2). Potrošnja po logici se odnosi na potrošnju logičkih elemenata čipa poput LUT-ova i flip-flopova. Potrošnja po signalima se odnosi na potrošnju po linijama i elementima za rutiranje (preciznije komutaciju) linija u čipu. Prikaz **By Clock Domain** daje prikaz potrošnje brzih linija takta u čipu po takt domenima (slika 3.3.11.3). Polje frekvencije takta *clk* je popunjeno vrednošću 100MHz, koja je uzeta iz fajla generisanog simulacijom. Kasnije u okviru ove sekcije će biti prikazano kako se u okviru ISim simulatora generiše fajl za upotrebu u analizi potrošnje. U narednim sekcijama će biti prikazano kako definisati frekvenciju takta u okviru .ucf fajla. **By Resource Type** daje prikaz po delovima dizajna u zavisnosti od njihovog tipa. U stvari, ovaj prikaz je identičan tabeli **On-Chip** iz **View-> Summary** prikaza. U podopcijama se dobija prikaz detaljnijih izveštaja - **Logic**, **Signals** i **IOs** prikazi.

Name	Power (W)	Frequency (MHz)	Buffer	Buffer Enable (%)	Enable Signal	Fanout	Slice Fanout
Clocks							
clk_BUFGP/IBUFG	0.00182	100.0	NA	NA	NA	5	2
clk	0.00000	0.0	IOB	NA	NA	NA	NA
clk_BUFGP/BUFG	0.00060	100.0	BUFG	NA	NA	NA	NA
clk_BUFGP/IBUFG	0.00004	100.0	IOB	NA	NA	1	1
clk_BUFGP	0.00119	100.0	BUFG	NA	NA	4	1
Total	**0.00182**						

Slika 3.3.11.3. – Potrošnja po takt domenima

U okviru **Logic** prikaza se dobija pregled upotrebljene logike u dizajnu i procena potrošnje po svakoj komponenti upotrebljene logike (slika 3.3.11.4).

Name	Power (W)	Type	Clock (MHz)	Clock Name	Signal Rate
brojac_inst/Mcount_q_int_xor<1>11	0.00002	B6LUT (SLICEL)	Async	Async	52.0
brojac_inst/Mcount_q_int_xor<0>11_INV_0	0.00002	A6LUT (SLICEL)	Async	Async	43.0
brojac_inst/Mcount_q_int_xor<2>11	0.00002	C6LUT (SLICEL)	Async	Async	43.0
displej_inst/Mrom_displej41	0.00002	A6LUT (SLICEL)	Async	Async	43.0
displej_inst/Mrom_displej31	0.00002	C6LUT (SLICEL)	Async	Async	42.5
brojac_inst/q_int_0	0.00001	AFF (SLICEL)	100.0	clk_BUFGP	42.5
brojac_inst/Mcount_q_int_xor<3>11	0.00001	D6LUT (SLICEL)	Async	Async	32.5
displej_inst/Mrom_displej11	0.00001	A6LUT (SLICEL)	Async	Async	26.5
displej_inst/Mrom_displej61	0.00001	D6LUT (SLICEL)	Async	Async	26.0
displej_inst/Mrom_displej51	0.00001	A6LUT (SLICEL)	Async	Async	25.5
displej_inst/Mrom_displej111	0.00001	A6LUT (SLICEL)	Async	Async	25.0
displej_inst/Mrom_displej21	0.00001	D6LUT (SLICEL)	Async	Async	18.5
brojac_inst/q_int_1	0.00001	BFF (SLICEL)	100.0	clk_BUFGP	17.0
brojac_inst/q_int_2	0.00000	CFF (SLICEL)	100.0	clk_BUFGP	8.5
brojac_inst/q_int_3	0.00000	DFF (SLICEL)	100.0	clk_BUFGP	8.0
Total	0.00016				

Slika 3.3.11.4. – Potrošnja po delovima logike

U okviru **Signals** prikaza dobija se zbirni prikaz potrošnje po **Data** i **Control** signalima. Pod **Control** signalima se podrazumevaju signali koji kontrolišu rad flip-flopova, set/reset i signal dozvole za takt. U okviru prikaza **Signals**, mogu se selektovati posebno **Data** prikaz i **Control** prikaz. **Data** prikaz daje pregled potrošnje po *data* signalima dizajna (slika 3.3.11.5). Kolona **Clock** definiše da li je signal asinhron ili deo nekog takt domena u kom slučaju se ispisuje takt koji kontroliše taj signal.

Name	Power (W)	Signal Rate	Fanout	Slice Fanout	Clock
brojac_inst/q_int<0>	0.00007	42.5	11	11	clk_BUFGP
displej_out_3_OBUF	0.00005	42.5	1	1	Async
displej_out_4_OBUF	0.00005	43.0	1	1	Async
displej_out_6_OBUF	0.00004	26.0	1	1	Async
displej_out_0_OBUF	0.00004	26.5	1	1	Async
displej_out_1_OBUF	0.00003	25.0	1	1	Async
displej_out_5_OBUF	0.00003	25.5	1	1	Async
brojac_inst/q_int<1>	0.00002	17.0	10	10	clk_BUFGP
displej_out_2_OBUF	0.00002	18.5	1	1	Async
brojac_inst/q_int<2>	0.00001	8.5	10	10	clk_BUFGP
brojac_inst/q_int<3>	0.00001	8.0	9	9	clk_BUFGP
Total	0.00039				

Slika 3.3.11.5. – Potrošnja po *data* signalima

U okviru **Control** prikaza dobija se zbirni prikaz potrošnje po **Clock Enable** i **Set/Reset** signalima. Oba tipa kontrolnih signala imaju svoje detaljne prikaze **Clock Enable**, odnosno **Set/Reset**.

IOs daje prikaz potrošnje po pinovima (portovima) top level dizajna. Između ostalog prikazuje se standard korišćen za signale koji se razmenjuju preko pinova sa okolinom FPGA čipa, pregled potrošnje po tipovima izvora napajanja, smerovi pinova i dr.

Kao što je već prethodno pomenuto, u analizi potrošnje snage se mogu koristiti fajlovi koji se generišu u okviru simulacije, a koji sadrže aktivnosti signala na osnovu kojih će se posle procenjivati potrošnja snage dizajna. Postoje dva tipa fajlova koji se mogu generisati - .saif i .vcd. Generisanje ovih fajlova se radi upisom odgovarajućih tekstualnih komandi u ISim simulatoru. Pri tome, može da se koristi funkcionalna ili vremenska simulacija. Za kvalitetniju procenu bolje je koristiti vremensku simulaciju, pa će u narednom primeru biti prikazano generisanje navedenih fajlova sa aktivnošću signala u slučaju vremenske simulacije. Na identičan način bi se generisali ti fajlovi i u okviru funkcionalne simulacije.

Izaberimo u strukturi projekta ISE aplikacije opciju **Simulation View** i u okviru nje **Post-Route** opciju da bismo mogli aktivirati vremensku simulaciju. Pokrenimo ISim simulator opcijom **Simulate Post-Place & Route Model** iz toka projekta. U donjem delu prozora kada je izabran tab **Console** možemo da ukucavamo komande za generisanje željenih fajlova aktivnosti signala. Sve komande koje se u nastavku navedu su *case-sensitive*. Za aktivaciju praćenja aktivnosti signala i kreiranje .saif fajla koristimo komandu:

saif open -*file* ime_fajla.saif -*allnets*

gde ime_fajla.saif predstavlja naziv fajla koji će biti kreiran/otvoren i koji će sadržati aktivnosti signala iz simulacije. Ako fajl već postoji on će biti prepisan novim vrednostima. Komanda **saif open** pokreće praćenje signala. Komanda **saif close** završava praćenje aktivnosti signala i zatvara fajl sa upisanim vrednostima aktivnosti signala. Ako se izostavi opcija -*fajl* ime_fajla.saif izostavi, koristiće se difolt naziv fajla. Opcija -*allnets* je neophodna da bi se posmatrala aktivnost svih signala u dizajnu, a ne samo portova top level entiteta. Ako želimo da snimimo aktivnosti signala u toku čitave simulacije, postupak za generisanje .saif fajla je sledeći:

1. Restartovati simulaciju klikom na opciju **Restart** u redu ikonica ispod glavnog menija ili aktivacijom opcije **Simulation->Restart** iz glavnog menija.
2. U konzoli ukucati komandu **saif open** -*file* ime_fajla.saif -*allnets*, pri čemu podesiti željeni naziv fajla.
3. Izvršiti simulaciju.
4. Po završetku simulacije sačuvati .saif fajl tako što se ukuca komanda **saif close** u konzoli.

Postoji i drugi način za generisanje .saif fajla. U okviru toka projekta ISE aplikacije se izabere pomenuta **Simulate Post-Place & Route Model** opcija i desnim klikom na nju se izabere **Process Properties** opcija. Aktivacijom opcije **Generate Saif File for Power Optimization/Estimation** u okviru **ISim Properties** podešavanja, ISim simulator će automatski generisati .saif fajl, pri čemu se naziv i lokacija fajla takođe podešavaju u okviru **ISim Properties** podešavanja. Navedena opcija je moguća samo za **Simulate Post-Place & Route Model**.

Za generisanje .vcd fajla se koristi sledeći postupak:

1. Restartovati simulaciju klikom na opciju **Restart** u redu ikonica ispod glavnog menija ili aktivacijom opcije **Simulation->Restart** iz glavnog menija.
2. U konzoli ukucati komandu **vcd dumpfile** ime_fajla.vcd, pri čemu podesiti željeni naziv fajla.
3. U konzoli ukucati komandu **vcd dumpvars** -*m* /ime_instance_top_level_entiteta, gde u ime_instance_top_level_entiteta treba staviti naziv instance komponente top level entiteta dizajna iz testbenča. Naziv instance top level entiteta se može lako videti i iz **Instance and Process Name** kolone sa leve strane ISim simulatora. Možemo staviti i naziv instance nekog drugog entiteta ako želimo da pratimo aktivnosti samo dela signala dizajna.
4. Izvršiti simulaciju.
5. Po završetku simulacije sačuvati .vcdfajl tako što se ukuca komanda **vcd dumpflush** u konzoli.

Navedene su samo osnovne komande za generisanje .saif i .vcd fajlova, dodatne komande se mogu naći u pomoćnoj dokumentaciji ISE aplikacije koja je dostupna preko **Help->Help Topics** iz glavnog menija ISE aplikacije.

Uključivanje .saif fajla u proces analize potrošnje snage se radi tako što se u toku projekta selektuje opcija **XPower Analyzer** i potom iz glavnog menija selektuje opcija **Process->Process Properties**. U okviru podešavanja **Load Simulation File** treba uneti lokaciju željenog .saif fajla, pri čemu postoji dugme **...** za brauzovanje do željenog fajla.

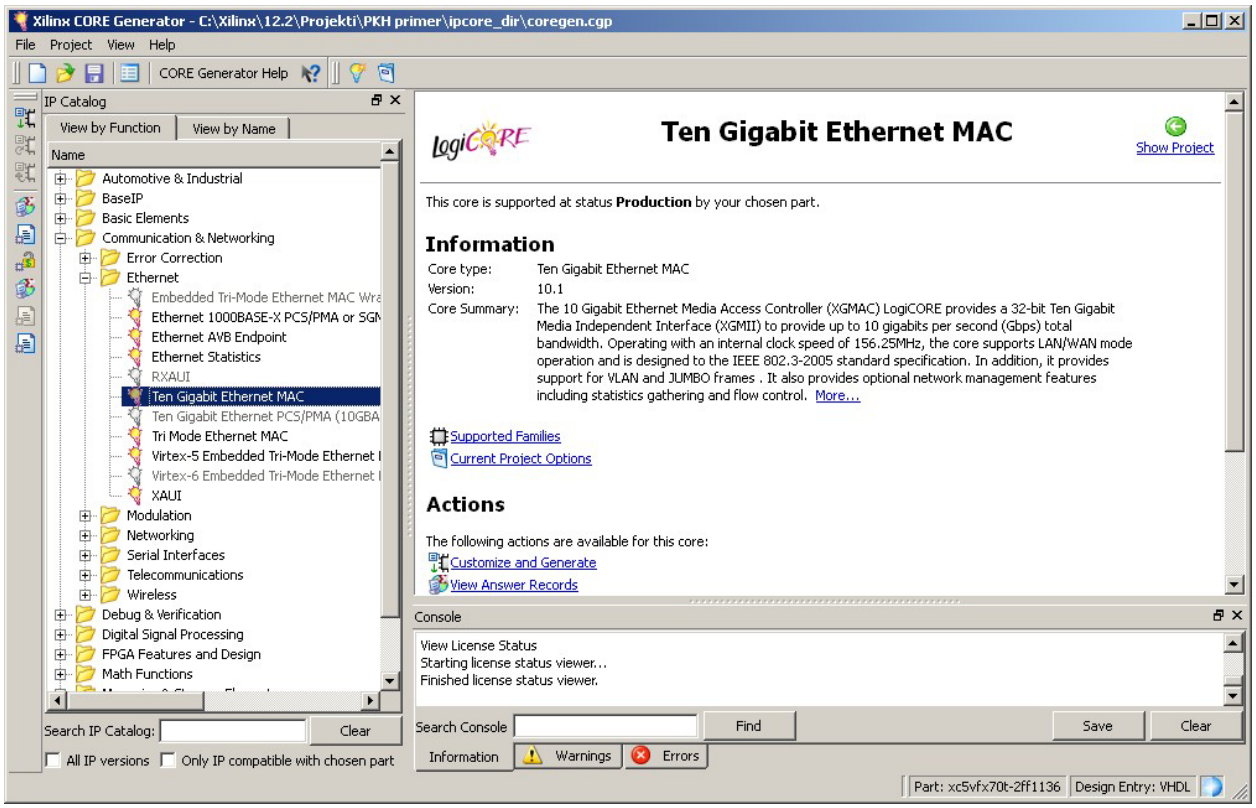
Uključivanje .vcd fajla se vrši tako što se u isto podešavanje **Load Simulation File** unese naziv .vcd fajla pri čemu treba da se taj fajl nalazi u radnom direktorijumu projekta, ili da se unese čitava ili relativna putanja dotičnog fajla ako se on nalazi na nekoj drugoj lokaciji. U ovom slučaju dugme za brauzovanje je nekorisno jer ono omogućava brauzovanje samo za .saif fajlove.

Napomenimo još da opcija **Generate Power Data** iz toka projekta, koja se nalazi odmah ispod **Xpower Analyzer** opcije, takođe vrši identičnu analizu potrošnje snage dizajna, sa razlikom da se generiše izveštaj u tekstualnom obliku (**Power Report**) koji se smešta uz ostale izveštaje procesa kompajliranja (grupa **Detailed Reports**). Podešavanja za opciju **Generate Power Data** se dobijaju selekcijom te opcije pa izborom **Process->Process Properties** iz glavnog menija.

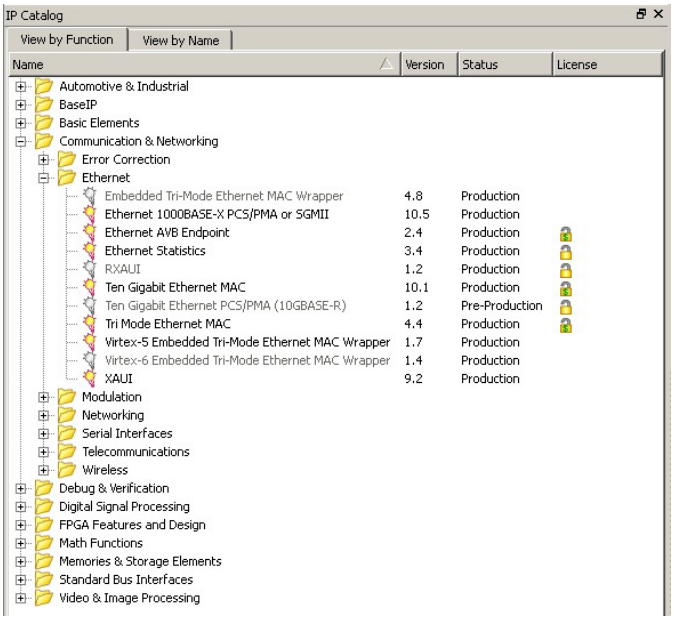
3.3.12. Upotreba megafunkcija

ISE softverski paket, slično kao i Quartus softverski paket ima na raspolaganju megafunkcije koje omogućavaju upotrebu specijalnih resursa FPGA čipa poput internih memorijskih blokova na jednostavan način. Isto tako, u megafunkcijama koje su na raspolaganju, se nalaze i često korišćene funkcionalnosti poput eternet kontrolera, PCI kontrolera i sl. U okviru ISE softverskog alata megafunkcije se nazivaju IP jezgra (*Intelectual Property Cores*), ili kako se češće u žargonu izgovara - IP korovi. Međutim, u nastavku teksta ćemo koristiti termin megafunkcija radi konzistentnosti.

Za otvaranje menija za izbor megafunkcije treba aktivirati opciju **Tools->Core Generator** iz glavnog menija. Po aktiviranju ove opcije otvara se prozor prikazan na slici 3.3.12.1. U koloni sa leve strane se nalazi prikaz megafunkcija koje su na raspolaganju, pri čemu se u vrhu kolone nalaze tabovi za izbor prikaza. **View by Function** prikaz sortira megafunkcije u logičke grupe, poput na primer grupe koja sadrži memorije i komponente za skladištenje (**Memories & Storage Elements**). **View by Name** sortira megafunkcije po njihovim nazivima. Megafunkcije čija imena su u sivoj boji nisu na raspolaganju. Pored naziva megafunkcije, postoje još tri kolone - verzija (**Version**), status (**Status**) i licenca (**License**) kao što je prikazano na slici 3.3.12.2. Polje verzije prikazuje trenutnu verziju megafunkcije, dok polje statusa daje informaciju o statusu megafunkcije. Na primer, status **Production** označava da je megafunkcija u potpunosti funkcionalna i testirana. Ukoliko je prazno polje licence onda je megafunkcija u potpunosti na raspolaganju korisniku. Ako postoji oznaka katanca u polju licence, onda dotična megafunkcija mora da se licencira. Licenca je besplatna ako ne postoji znak dolara u katancu, u suprotnom licenca mora da se plati. Status licence se može videti desnim klikom na naziv megafunkcije i potom treba izabrati opciju **View License Status**.



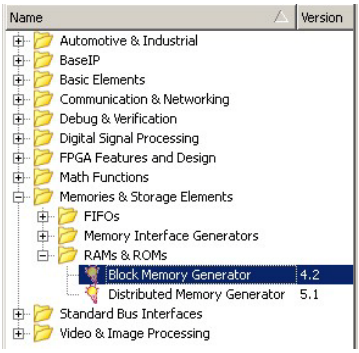
Slika 3.3.12.1. – Core Generator prozor



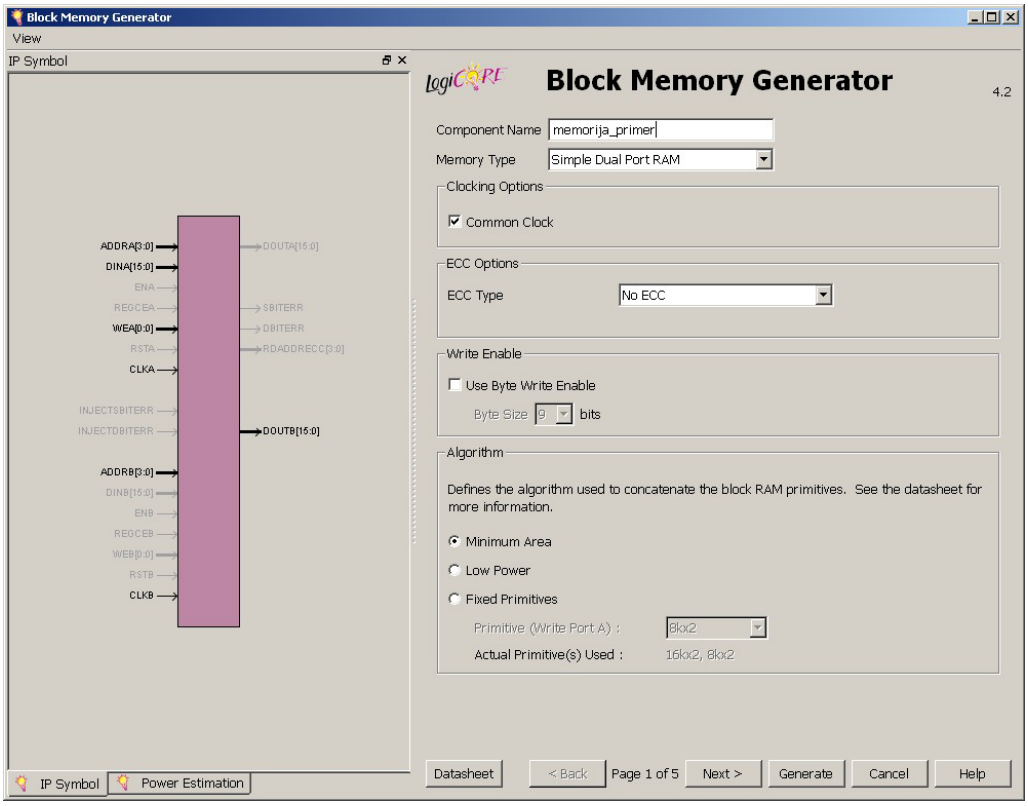
Slika 3.3.12.2. – Informacije o verziji, statusu i licenci megafunkcije

Po difoltu se prikazuju samo poslednje verzije megafunkcija. Ako želimo da vidimo sve dostupne verzije megafunkcija, treba u dnu prozora sa leve strane aktivirati opciju **All IP versions**. U dnu prozora se takođe nalazi opcija **Only IP compatible with chosen part** čijom aktivacijom se prikazuju samo one megafunkcije koje su kompatibilne sa izabranim FPGA čipom.

Osnovne informacije o selektovanoj megafunkciji se prikazuju u desnom delu prozora (slika 3.3.12.1). Ispod informacija o megafunkciji se nalazi lista akcija koje se mogu preduzeti. **View Answer Record** vodi na veb stranicu koja sadrži listu pitanja i odgovora vezanih za dotičnu megafunkciju. **View Data Sheet** otvara dokumentaciju megafunkcije. Uvek je poželjno pre prve upotrebe pročitati dokumentaciju megafunkcije. **View License Status** daje informacije o statusu licence za dotičnu megafunkciju. **View Product Webpage** otvara veb stranicu koja daje informacije o megafunkciji. **New Version Information** daje informacije o dotičnoj verziji - koji su bagovi otklonjeni, koje su nove funkcionalnosti dodate i sl. Opcija **Customize and Generate** pokreće proces kreiranja primerka megafunkcije. Napomenimo da nisu sve opcije uvek prisutne. Na primer, opcija **View License Status** nije prisutna ako je megafunkcija u potpunosti na raspolaganju korisniku i ne zahteva upotrebu licence. Takođe ako megafunkcija nije na raspolaganju (naziv megafunkcije je obeležen sivom bojom) tada na raspolaganju neće biti opcija **Customize and Generate**.



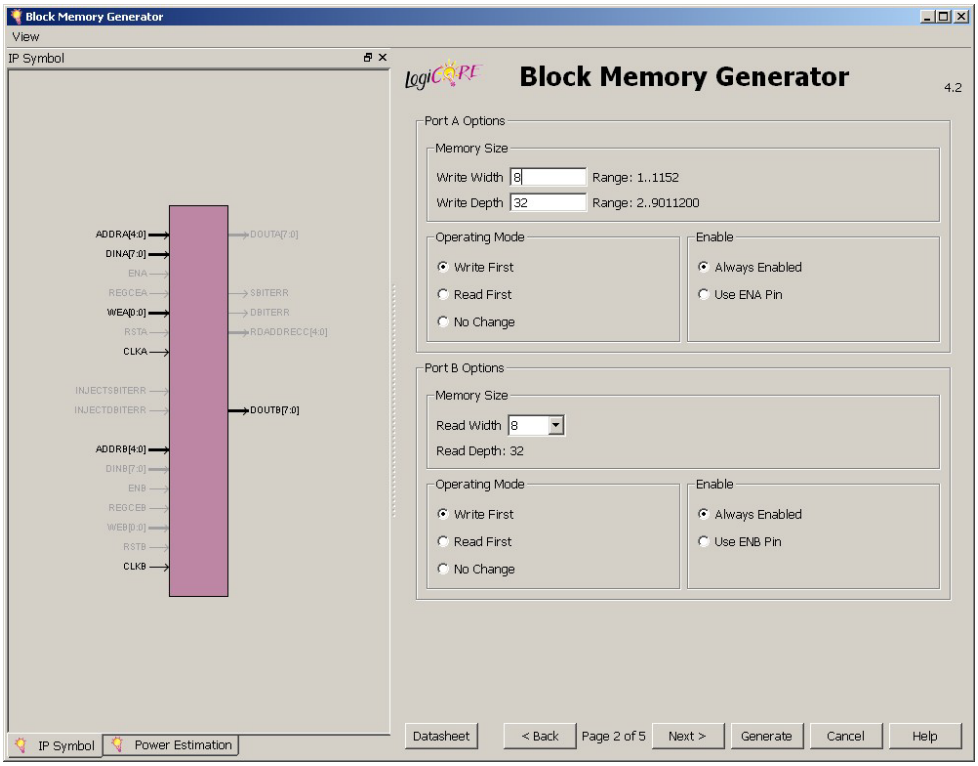
Slika 3.3.12.3. – Izbor megafunkcije Block Memory Generator



Slika 3.3.12.4. – Prvi prozor u podešavanju interne memorije

Kreirajmo jednu dvoportnu memoriju kao što smo uradili i kod opisa megafunkcija Quartus aplikacije. Selektujmo megafunkciju **Block Memory Generator** kao što je prikazano na slici 3.3.12.3. I zatim u desnom delu prozora aktivirajmo opciju **Customize and Generate** da bismo pokrenuli proces kreiranja jednog primerka dvoportne memorije.

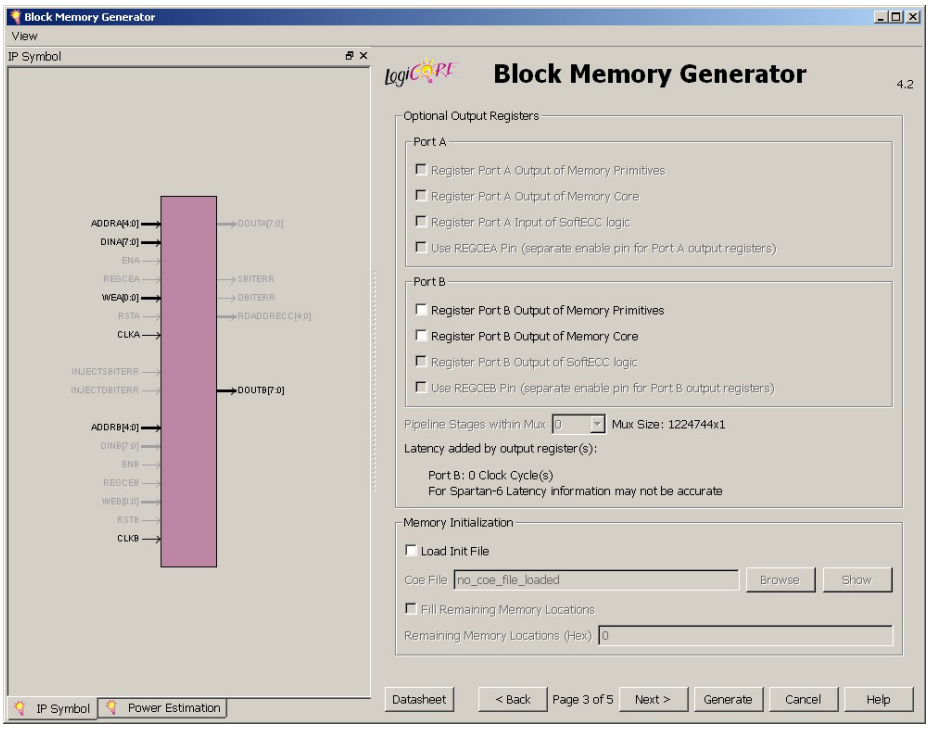
U prvom otvorenom prozoru prikazanom na slici 3.3.12.4, podešava se tip memorije koju realizujemo. Selektujemo **Simple Dual Port RAM** memoriju u padajućem meniju **Memory Type**. U polje **Component Name** upisujemo željeno ime kreirane memorije (preciznije entiteta memorije). U prikazu memorije sa leve strane je prikazana memorija se trenutnim podešavanjima, pa na jednostavan način možemo videti strukturu svih tipova memorije iz padajućeg menija **Memory Type** i samim tim izabrati onaj tip memorije koji nam najviše odgovara. U slučaju sa slike 3.3.12.4 port A se koristi za upis, a port B za čitanje. Ostale opcije ostavimo na difolt vrednostima. Dugme **Datasheet**, koje se nalazi u dnu ekrana, otvara dokumentaciju megafunkcije gde su objašnjena sva podešavanja i detalji dotične megafunkcije što olakšava konfigurisanje megafunkcije. Dugme **Generate** odmah pokreće generisanje primerka megafunkcije, pri čemu su na svim preostalim prozorima ostavljena difolt podešavanja ako je u pitanju kreiranje novog primerka megafunkcije, odnosno stara podešavanja ako je u pitanju modifikacija postojećeg primerka megafunkcije. U dnu strane je napisan i redni broj prozora kao i ukupan broj prozora u okviru podešavanja ove megafunkcije. Izaberimo opciju **Next** da bi prešli na sledeća podešavanja.



Slika 3.3.12.5. – Drugi prozor u podešavanju interne memorije

Drugi prozor omogućava podešavanje broja lokacija (**Write/Read Depth**) i širine lokacija (**Write/Read Width**) za portove memorije (slika 3.3.12.5). U okviru **Enable** podešavanja se dodatno može podesiti da li je port uvek aktivan ili se koristi signal dozvole. Ako se koristi signal dozvole, tada port obavlja svoje aktivnosti samo kad je signal dozvole aktivan, u suprotnom je port ’zamrznut’ i ne može da obavlja ni funkcije upisa, ni čitanja, ni reseta.

Operating mode podešavanje određuje redosled operacija u slučaju istovremenog upisa i čitanja identične lokacije.

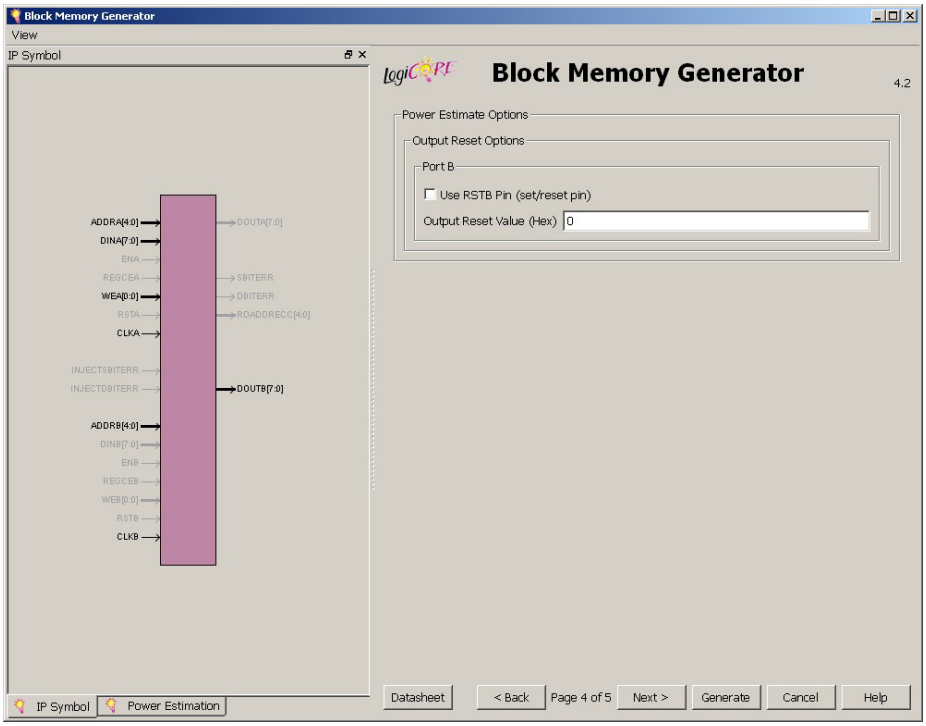


Slika 3.3.12.6. – Treći prozor u podešavanju interne memorije

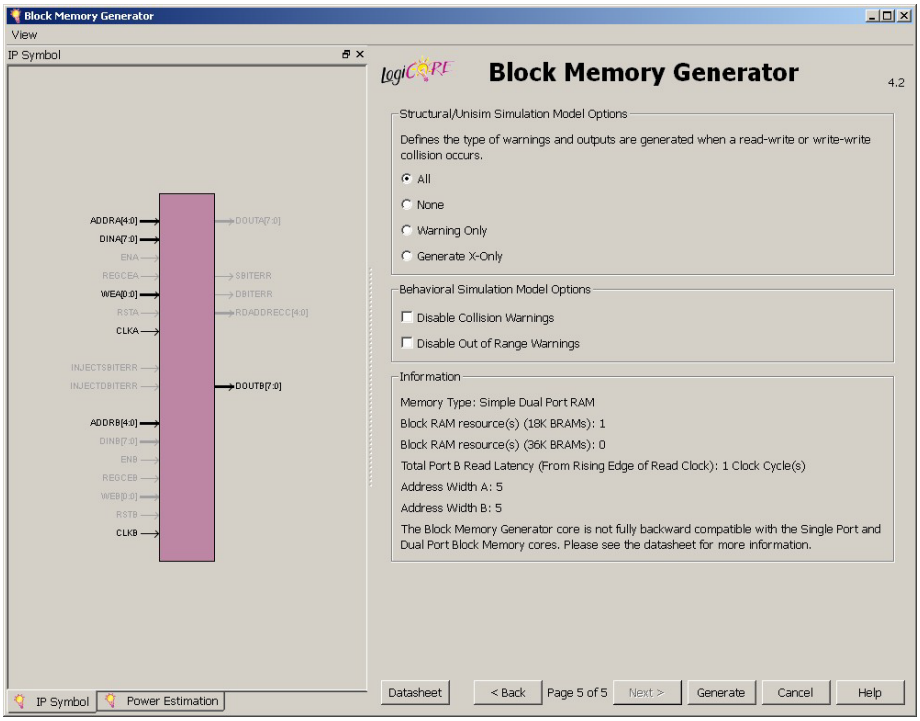
Treći prozor (slika 3.3.12.6) omogućava podešavanje izlaznih registara kod portova kod kojih je omogućena operacija čitanja (u prikazanom primeru u pitanju je samo port B). Aktivacija opcije **Register Port B Output of Memory Primitives** dodaje registar na izlazu same memorijske primitive, dok **Register Port B Output of Memory Core** dodaje registar iza multipleksera kompletnog memorijskog bloka. Cilj oba registra je da poboljšaju performanse dizajna sa stanovišta postizanja veće maksimalne frekvencije dizajna, ali svaki od registara dodaje jedan takt kašnjenja. U slučaju aktivacije bar jednog izlaznog registra dobija se i opcija dodavanja signala dozvole za kreirane registre. Detaljnije informacije o ovim registrima i njihovoj poziciji se mogu naći u pomoćnoj dokumentaciji koja se otvara klikom na dugme **Datasheet**. U okviru ovog prozora se može uključiti i fajl koji sadrži inicijalni sadržaj memorije, pri čemu se tada aktivira i opcija upisa istog konstantnog sadržaja u sve lokacije koje nisu pokrivene fajlom. U sledećoj sekciji ćemo videti kako se kreira .coe fajl koji definiše inicijalni sadržaj memorije.

Na slici 3.3.12.7 je prikazan četvrti prozor u podešavanjima kreirane memorije. Ovaj prozor omogućava dodavanje reseta za portove koji se koriste za čitanje (port B u našem primeru). Tada se pomoću signala reseta porta može resetovati izlazni podatak iz memorije, pri čemu se u okviru ovog prozora definiše i vrednost koja se prikazuje na aktivirani reset signal.

Poslednji prozor u podešavanju daje mogućnost podešavanja simulacionog modela kreirane memorije (slika 3.3.12.8). Ovde se definiše način prijavljivanja kolizije (istovremenog upisa i čitanja iste lokacije) u okviru simulacije, a na raspolaganju je i mogućnost deaktivacije pojedinih tipova upozorenja u okviru simulacije.

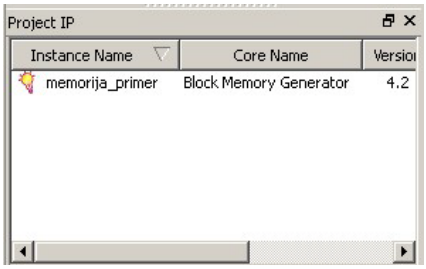


Slika 3.3.12.7. – Četvrti prozor u podešavanju interne memorije



Slika 3.3.12.8. – Poslednji prozor u podešavanju interne memorije

Klikom na dugme **Generate** se kreira konfigurisani primerak megafunkcije **Block Memory Generator**. Kreirani primerak megafunkcije se dodaje u listu kreiranih megafunkcija, pri čemu se lista nalazi ispod kolone sa prikazom megafunkcija sa leve strane **Core Generator** prozora (slika 3.3.12.9).



Slika 3.3.12.9. – Lista kreiranih primeraka megafunkcija

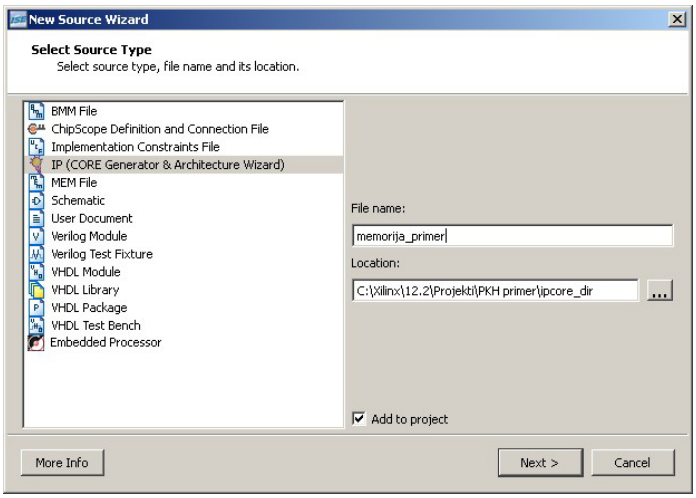
Nakon što smo kreirali željenu megafunkciju (odnosno njen primerak), potrebno ga je uključiti u projekat. Prvi način je uključivanje .xco fajla koji predstavlja kreirani primerak megafunkcije i samim tim svih fajlova koji su kreirani prilikom kreiranja dotične megafunkcije. Drugi način je dodavanje svih .vhd fajlova kreiranog primerka megafunkcije (koji su kreirani prilikom kreiranja megafunkcije) i eventualnih .ucf fajlova ako je megafunkcija kreirala i neka korisnička ograničenja poput lokacija pinova na koja treba da se vežu portovi kreirane megafunkcije. U slučaju kreiranja interne memorije nema .ucf fajlova, već samo jedan .vhd fajl. Koji metod koristiti je u principu svejedno i zavisi od stila dizajnera. Ako se koristi .xco fajl tada se dvostrukim klikom na .xco fajl otvara početni prozor konfiguracije dotičnog primerka megafunkcije i može se jednostavno promeniti konfiguracija megafunkcije. Da bi se instancirala megafunkcija u okviru projekta, potrebno je instancirati entitet megafunkcije u vidu komponente. Mana .xco fajla je što .vhd fajlovi megafunkcije nisu vidljivi iz strukture projekta. Ako se koriste .vhd fajlovi tada se na jednostavan način može iz same strukture projeka pristupati sadržajima tih .vhd fajlova. I u ovom slučaju instanciranje megafunkcije u okviru projekta je isto i zasniva se na instanciranju entiteta megafunkcije u vidu komponente. Mana ovog pristupa je što se modifikovanje megafunkcije mora raditi van projekta upotrebom **Core Generator** alata. U slučaju kada megafunkcija sadrži veći broj .vhd fajlova, važno je pročitati dokumentaciju megafunkcije da bi se odredila hijerarhija generisanih .vhd fajlova i time odredilo koji od tih .vhd entiteta treba instancirati kao komponentu. Struktura .vhd fajla koji predstavlja kreiranu memoriju je:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- synthesis translate_off
Library XilinxCoreLib;
-- synthesis translate_on
ENTITY memorija_primer IS
        port (
        clka: IN std_logic;
        wea: IN std_logic_VECTOR(0 downto 0);
        addra: IN std_logic_VECTOR(4 downto 0);
        dina: IN std_logic_VECTOR(7 downto 0);
        clkb: IN std_logic;
        addrb: IN std_logic_VECTOR(4 downto 0);
        doutb: OUT std_logic_VECTOR(7 downto 0));
END memorija_primer;
ARCHITECTURE memorija_primer_a OF memorija_primer IS
-- synthesis translate_off
component wrapped_memorija_primer
        port (
        clka: IN std_logic;
        wea: IN std_logic_VECTOR(0 downto 0);
        addra: IN std_logic_VECTOR(4 downto 0);
        dina: IN std_logic_VECTOR(7 downto 0);
```

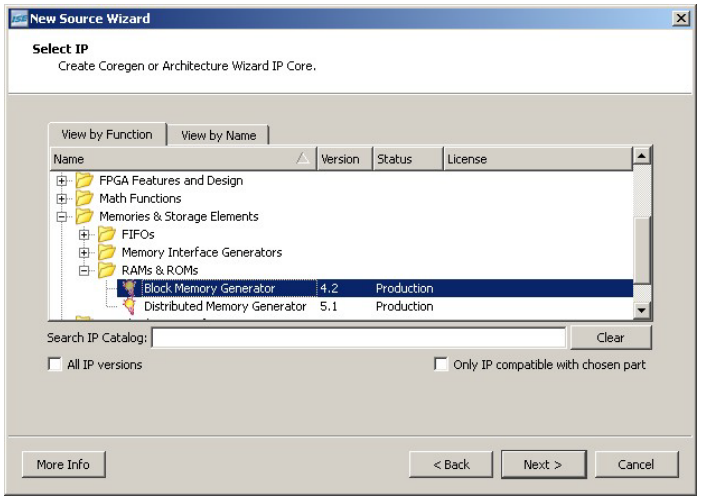
```
            clkb: IN std_logic;
            addrb: IN std_logic_VECTOR(4 downto 0);
            doutb: OUT std_logic_VECTOR(7 downto 0));
end component;
-- Configuration specification
            for all : wrapped_memorija_primer use entity XilinxCoreLib.blk_mem_gen_v4_2(behavioral)
                  generic map(
                              c_has_regceb => 0,
                              c_has_regcea => 0,
                              c_mem_type => 1,
                              c_rstram_b => 0,
                              c_rstram_a => 0,
                              c_has_injecterr => 0,
                              c_rst_type => "SYNC",
                              c_prim_type => 1,
                              c_read_width_b => 8,
                              c_initb_val => "0",
                              c_family => "virtex5",
                              c_read_width_a => 8,
                              c_disable_warn_bhv_coll => 0,
                              c_use_softecc => 0,
                              c_write_mode_b => "WRITE_FIRST",
                              c_init_file_name => "no_coe_file_loaded",
                              c_write_mode_a => "WRITE_FIRST",
                              c_mux_pipeline_stages => 0,
                              c_has_softecc_output_regs_b => 0,
                              c_has_mem_output_regs_b => 0,
                              c_has_mem_output_regs_a => 0,
                              c_load_init_file => 0,
                              c_xdevicefamily => "virtex5",
                              c_write_depth_b => 32,
                              c_write_depth_a => 32,
                              c_has_rstb => 0,
                              c_has_rsta => 0,
                              c_has_mux_output_regs_b => 0,
                              c_inita_val => "0",
                              c_has_mux_output_regs_a => 0,
                              c_addra_width => 5,
                              c_has_softecc_input_regs_a => 0,
                              c_addrb_width => 5,
                              c_default_data => "0",
                              c_use_ecc => 0,
                              c_algorithm => 1,
                              c_disable_warn_bhv_range => 0,
                              c_write_width_b => 8,
                              c_write_width_a => 8,
                              c_read_depth_b => 32,
                              c_read_depth_a => 32,
                              c_byte_size => 9,
                              c_sim_collision_check => "ALL",
                              c_common_clk => 1,
                              c_wea_width => 1,
                              c_has_enb => 0,
                              c_web_width => 1,
                              c_has_ena => 0,
                              c_use_byte_web => 0,
                              c_use_byte_wea => 0,
                              c_rst_priority_b => "CE",
                              c_rst_priority_a => "CE",
                              c_use_default_data => 0);
```

```
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_memorija_primer
                port map (
                        clka => clka,
                        wea => wea,
                        addra => addra,
                        dina => dina,
                        clkb => clkb,
                        addrb => addrb,
                        doutb => doutb);
-- synthesis translate_on
END memorija_primer_a;
```

Postoji i drugi način za započinjanje kreiranja megafunkcije. Naime, opcijom **Project-> New Source** iz glavnog menija ISE aplikacije otvara se prozor za izbor tipa izvorišnog fajla koji kreiramo (slika 3.3.12.10). Izabere se opcija **IP (Core Generator & Architecture Wizard)** i u polje **File name** se upiše ime primerka megafunkcije. Opcija **Add to project** u dnu prozora je po difoltu aktivna i označava da će se kreirana megafunkcija dodati u projekat. Ako to ne želimo onda treba dekativirati tu opciju.



Slika 3.3.12.10. – Izbor Core Generator tipa kao izvorišnog fajla



Slika 3.3.12.11. – Prozor za izbor željene megafunkcije

Na slici 3.3.12.11 je prikazan prozor za izbor željene megafunkcije. Ovaj prozor je u stvari skraćeni prikaz **Core Generator** prozora sa slike 3.3.12.1. Nakon izbora željene megafunkcije i klika na dugme **Next** otvara se prozor sa rezimeom izbora gde je navedena lokacija gde će biti snimljen kreirani primerak megafunkcije kao i naziv i tip kreirane megafunkcije. Klikom na dugme **Finish** otvara se početni prozor za podešavanje izabrane megafunkcije. U ovom slučaju nije moguće menjati ime kreiranog primerka megafunkcije jer je on već podešen na prozoru prikazanom na slici 3.3.12.10. I ovde važe identične napomene vezane za uključivanje megafunkcije u projekat. Ako je bila neaktivna opcija **Add to project** onda možemo dodati kreiranu funkciju bilo dodavanjem .xco fajla, bilo dodavanjem .vhd i eventualnih .ucf fajlova. Ako je opcija **Add to project** bila aktivna onda je po difoltu u projekat dodan .xco fajl. Ako ipak želimo da dodamo .vhd i .ucf fajlove, onda treba prvo ukloniti .xco fajl iz strukture projekta, a potom dodati u projekat .vhd i .ucf fajlove megafunkcije.

Navedimo da slične napomene o dodavanju megafunkcije u projekat važe i za Quartus aplikaciju. Tamo je po difoltu dodavan .qip fajl koji je ekvivalent .xco fajla. I kod Quartus aplikacije se može ukloniti .qip fajl iz projekta i potom dodati .vhd fajlovi (i eventualni drugi izvorišni fajlovi poput Verilog fajlova) megafunkcije u projekat.

3.3.13. Kreiranje inicijalnog sadržaja internih memorije

Za kreiranje inicijalnog sadržaja memorije se koristi .coe fajl. Pored kreiranja inicijalnog sadržaja interne memorije, .coe fajlovi se mogu koristiti u inicijalizaciji i podešavanjima još nekih megafunkcija (na primer u megafunkciji koja kreira FIR filtar, .coe fajl se koristi za podešavanje koeficijenata FIR filtra). Ovaj fajl se može kreirati u bilo kom tekstualnom editoru. Na početku ovog fajla se definiše format zapisa koji će se koristiti. Postoje dve ključne reči: MEMORY_INITIALIZATION_RADIX za definisanje formata .coe fajla koji se koristi za inicijalizaciju sadržaja memorije i RADIX za definisanje formata u svim ostalim slučajevima. Primer:

```
MEMORY_INITIALIZATION_RADIX = 16;
```

Vrednost koja se dodeljuje predstavlja format koji se koristi u zapisu. Na primer, vrednost 16 označava heksadecimalni format, a vrednost 2 označava binarni format. Komentari u .coe fajlovima se pišu iza ; i vrede do kraja tekuće linije. Ako linija sadrži samo komentar tada je na početku linije znak ; i iza njega ide tekst komentara. Ključne reči za dodelu vrednosti su: MEMORY_INITIALIZATION_VECTOR, PATTERN, COEFDATA i dr.. U slučaju definisanja inicijalnog sadržaja interne memorije koristi se MEMORY_INITIALIZATION_VECTOR. U slučaju definisanja koeficijenata FIR filtra bi se koristio COEFDATA. U dokumentaciji megafunkcija koje imaju mogućnost upotrebe .coe fajlova se nalaze podrobnije informacije o načinu dodele vrednosti odgovarajućim parametrima megafunkcije (na primer, kod interne memorije u pitanju je inicijalni sadržaj memorije). Poželjno je uvek pročitati dokumentaciju megafunkcije (deo koji se odnosi na strukturu .coe fajla za tu megafunkciju) za koju kreiramo .coe fajl. Iza ključne reči za dodelu vrednosti seređaju vrednosti lokacija/parametara i iza poslednje vrednosti se stavlja tačka-zarez. Primer definisanja inicijalnog sadržaja memorije koja ima 16 lokacija, pri čemu je širina lokacije 8 bita:

```
memory_initialization_radix=16;
memory_initialization_vector=
ff,
ab,
```

```
f0,
11,
11,
00,
01,
aa,
bb,
cc,
dd,
ef,
ee,
ff,
00,
ff;
```

Kao što vidimo prvo je definisan heksadecimalni format zapisa, a potom su redom navedene vrednosti svih 16 lokacija. Sadržaji lokacija su razdvojeni zarezom, a iza sadržaja poslednje lokacije se stavlja tačka-zarez. Vrednosti lokacija se mogu navesti i manjem broju redova:

```
memory_initialization_radix=16;
memory_initialization_vector=
ff, ab, f0, 11, 11, 00, 01, aa, bb, cc, dd, ef, ee, ff, 00, ff;
```

3.3.14. Definisanje .ucf fajlova

U ISE okruženju .ucf fajlovi se koriste za definisanje korisničkih ograničenja, poput dodele pinova čipa portovima top level entiteta, definisanja frekvencije takta koja treba da se postigne i dr. Otuda i naziv ekstenzije ucf (*user constraints file*). U okviru projekta može da koristi i više .ucf fajlova.

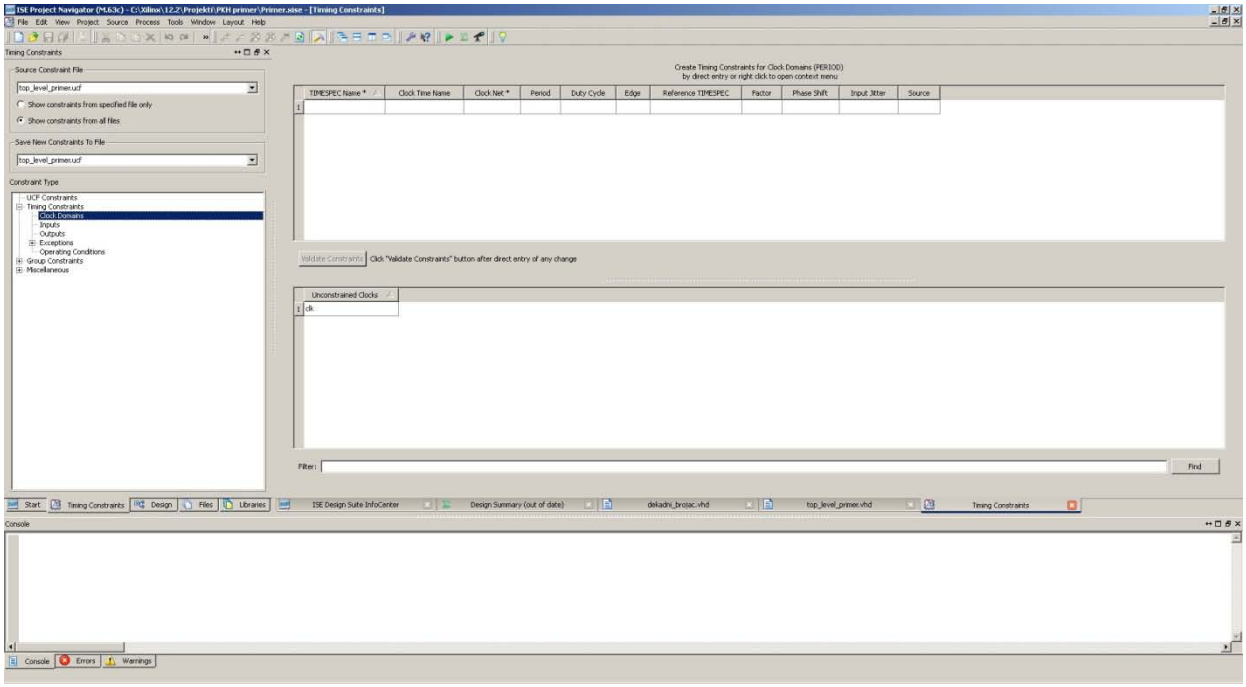
Postoje različiti tipovi korisničkih ograničenja koja se mogu definisati poput LOC ograničenja (definisanje lokacije dela dizajna), PERIOD ograničenja (definisanje periode takta), MAX_FANOUT ograničenja (definisanje maksimalnog broja izlaznih linija nekog elementa poput flip-flopa), i dr. Detaljnije informacije o svim ograničenjima i načinima njihove upotrebe mogu se naći u pomoćnoj dokumentaciji ISE aplikacije koja se otvara izborom opcije **Help-> Help Topics** iz glavnog menija.

Pomoću običnog tekstualnog editora može da se kreira .ucf fajl koji se potom uključuje u projekat i na ovaj način mogu da se kreiraju svi mogući tipovi korisničkih ograničenja. Kreiranje vremenskih ograničenja je moguće izborom **Tools->Constraints Editor** opcije iz glavnog menija. Kreiranje lokacijskih ograničenja je moguće izborom **Tools->PlanAhead** opcije iz glavnog menija.

U slučaju upotrebe tekstualnog editora, direktno se ukucavaju korisnička ograničenja. Na primer, ako želimo signalu reseta iz kreiranog projekta primera da dodelimo pin AJ24 koristimo izraz:

```
NET "reset" LOC = "AJ24";
```

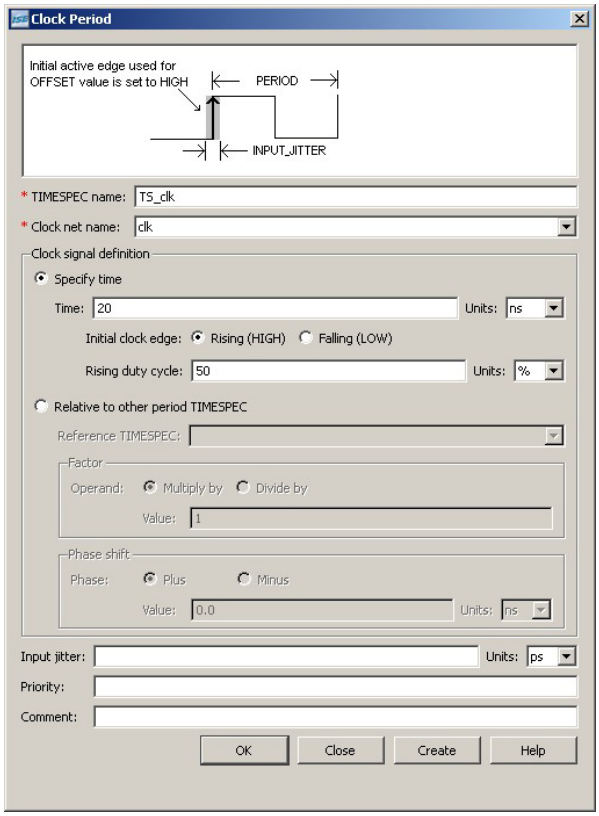
Očigledno svaki izraz mora da se završi sa znakom tačka-zarez. Ključna reč NET definiše da je u pitanju signal, a LOC predstavlja korisničko ograničenje kojem se dodeljuje naziv pina na koji se želi povezati signal *reset*.



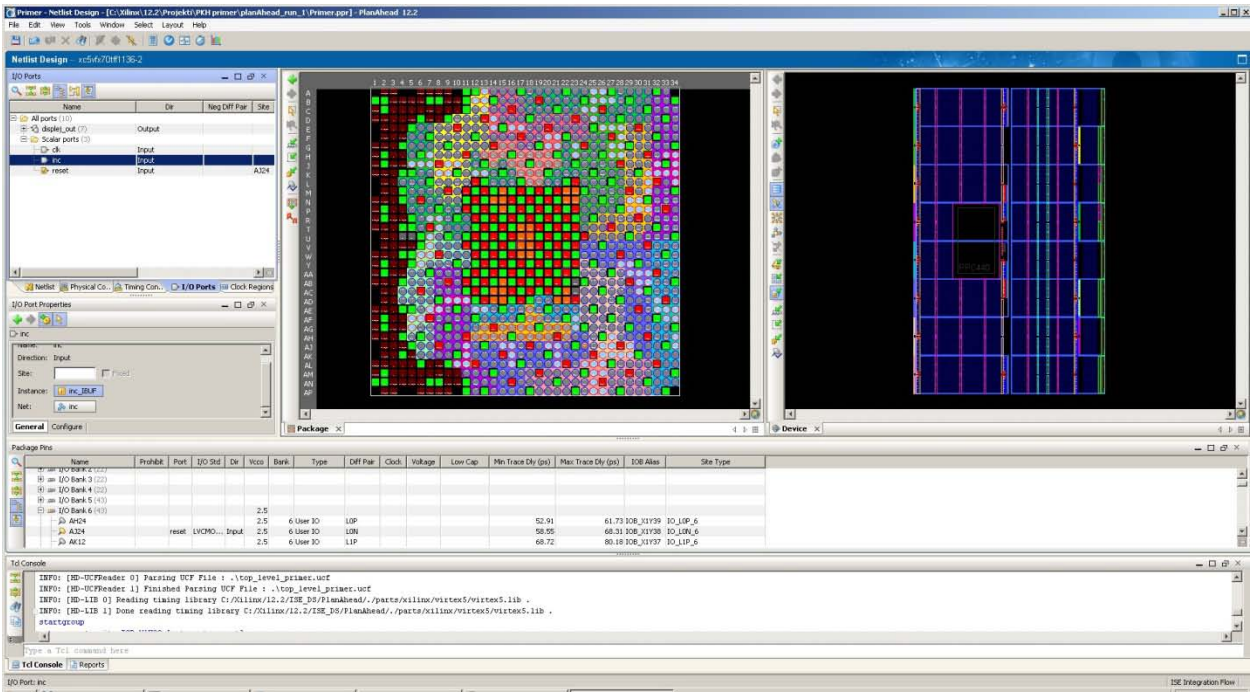
Slika 3.3.14.1. – Constraints Editor

Ako aktiviramo već pomenuti **Constraints Editor** otvara se prozor sa kreiranim praznim .ucf fajlom (na eventualno pitanje da li da se kreira .ucf fajl treba potvrdno odgovoriti da bi se otvorio ovaj editor). **Constraints Editor** je prikazan na slici 3.3.14.1. U **Constraint Type** prozoru sa leve strane biramo tip ograničenja. Na primer, izborom opcije **Clock Domains** se dobija prikaz svih taktova dizajna (slika 3.3.14.1). Desnim klikom na takt u listi **Unconstrained Clocks** otvara se meni u kome biramo opciju **Create Constraint** čime se otvara prozor za podešavanje parametara takta prikazan na slici 3.3.14.2. Kada se podese željeni parametri takta klikom na **OK** ili **Create** se kreira korisničko ograničenje za izabrani takt. Važnija podešavanja su **Time** gde se definiše perioda takta, kao i **Rising duty cycle** koje definiše odnos trajanja vrednosti ’1’ i ’0’ nivoa takta. Takođe se može definisati i relativni odnos prema nekom drugom taktu (ako naravno takav takt postoji i ako između njih postoji određena zavisnost). Takođe se može definisati i ulazni džiter takta. Izborom opcije **UCF Constraints** u **Constraint Type** prozoru prikazuju se sva kreirana korisnička ograničenja. Izborom opcije **Timing Constraints** u **Constraint Type** prozoru prikazuju se sva kreirana korisnička vremenska ograničenja pri čemu se uz ograničenje navodi i tip ograničenja. Izborom opcija **Inputs** i **Outputs** u okviru **Timing Constraints** se dobija mogućnost definisanja da li su ulazni i izlazni portovi sinhronisani na neki takt dizajna i podešavanje parametara u slučaju da jesu. Ostala ograničenja neće biti objašnjena u okviru ovih skripti, a više o njima se može pročitati u pomoćnoj dokumentaciji ISE aplikacije. Primer kreiranog korisničkog ograničenja za signal takta kojim se definiše da je željena frekvencija 50MHz i da je odnos vrednosti ’0’ i ’1’ nivoa takta 50% (ove linije se mogu kreirati i tekstualnim editorom, bez upotrebe **Constraints Editor** alata):

```
NET "clk" TNM_NET = clk;
TIMESPEC TS_clk = PERIOD "clk" 20 ns HIGH 50%;
```



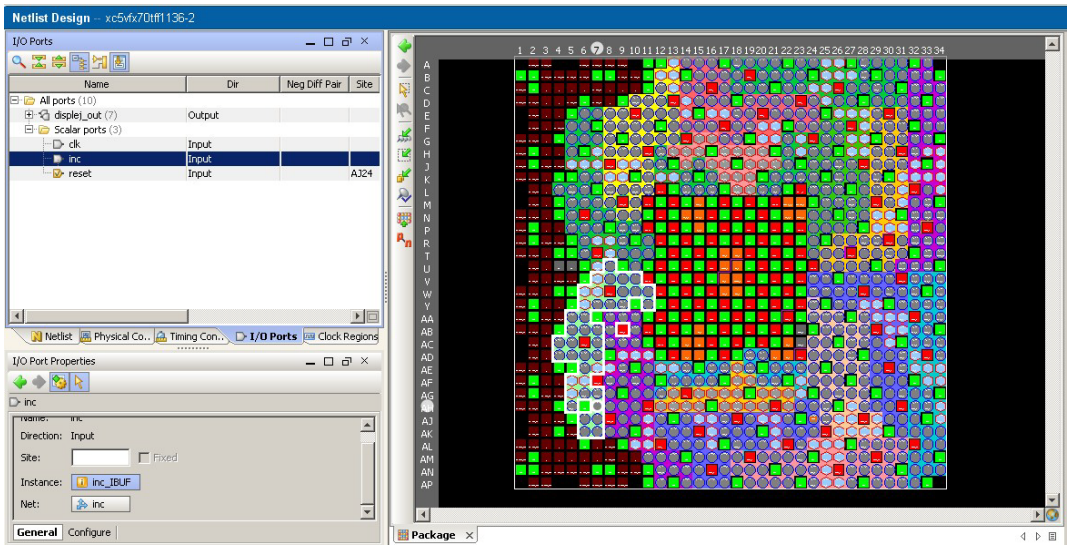
Slika 3.3.14.2. – Podešavanje parametara takta



Slika 3.3.14.3. – PlanAhead alat - podešavanje pinova

Aktivacijom opcije **Tools->PlanAhead->I/O Pin Planning** (jednom od dve ponuđene) iz glavnog menija se aktivira alat kojim se mogu izvršiti dodele pinova čipa portovima top level entiteta (slika 3.3.14.3). Razlika u dve varijante je u tome da li se uzima dizajn pre izvršenog

procesa sinteze ili posle. U meniju (koloni) sa leve strane nalazi se prozor sa opcijama koje se mogu podešavati, a koje se biraju sa tabovima na dnu prozora. Što se tiče dodele pinova bira se tab **I/O Ports** čime se dobija lista portova top level entiteta. Selekcijom nekog porta dobijamo mogućnost dodele pina izabranom portu ispod kolone koja prikazuje listu portova top level entiteta. U polje **Site** možemo upisati željeni pin. Po upisu željenog pina treba kliknuti na dugme **Apply** koje se pojavi po upisu lokacije pina na koji želimo povezati selektovani port top level entiteta. Nakon toga se u koloni **Site** dotičnog porta pojavljuje vrednost dodeljenog pina. I ovde važi napomena da sva ograničenja koja se kreiraju na ovaj način mogu da se kreiraju i običnim tekstualnim editorom. Dodelu pina putem tekstualnog editora smo već prikazali ranije u okviru ove sekcije.



Slika 3.3.14.4. – PlanAhead alat - podešavanje selektovanog pina

Aktivacijom opcije **Tools->PlanAhead->FloorPlan Area/IO//Logic** iz glavnog menija se otvara **PlanAhead** alat kojim je moguće definisati prostorna ograničenja dizajna. Na primer, pojedine delove dizajna je moguće pomeriti na željene lokacije u čipu i to definisati kao korisničko ograničenje tako da se kompajler forsira da tako definisane delove dizajna smesti na zadate lokacije. I ovde, naravno, važi napomena da sva ograničenja koja se kreiraju na ovaj način mogu da se kreiraju i običnim tekstualnim editorom.

3.3.15. Kreiranje particija

Particije se mogu kreirati na dva načina: kreiranjem .pxml fajla koji sadrži definicije particija, ili upotrebom **PlanAhead** alata. U okviru ovih skripti biće opisan prvi način kreiranja particija. Pri tome napomenimo da particije ne mogu da se definišu za sve familije Xilinx čipova. Na primer, u ISE 12.2 paketu particije nisu podržane za Virtex5 familiju, ali jesu za Viretx6 familiju.

Kreiraćemo dizajn sa dve particije, gde će dekadni brojač činiti jednu particiju dizajna, a displej drugu particiju dizajna. Sadržaj xpartition.pxml fajla je sledeći:

```
<?xml version="1.0" encoding="UTF-8" ?>
        <Project FileVersion="1.2" Name="Primer" ProjectVersion="2.0">
                <Partition Name="/top_level_primer" State="implement" ImportLocation="NONE">
                        <Partition Name="/top_level_primer/brojac_inst" State="implement" ImportLocation="NONE">
                        </Partition>
```

```
        <Partition Name="/top_level_primer/displej_inst" State="implement" ImportLocation="NONE">
        </Partition>
    </Partition>
</Project>
```

Kao što se vidi sadržaj fajla je napisan primenom xml formata, pri čemu je važno napomenuti da naziv fajla mora da bude *xpartition.pxml*. Prva linija definiše verziju xml standarda, kao i format kodiranja teksta. **Project** tag definiše verzije **FileVersion** i **ProjectVersion** čije vrednosti ne treba menjati, i naziv projekta **Name** koji treba postaviti na naziv projekta za koji pravimo particije (u našem slučaju projekat se zove *Primer* kao što je i navedeno na slici 3.3.2.1). Nakon toga se definišu particije u okviru **Partition** tagova. Važno je napomenuti da vrhovna particija mora biti top level entitet. Unutar vrhovne particije se navode druge particije, u našem primeru dekadni brojač i displej. Može se kreirati i veći broj hijerarhijskih nivoa particija metodom gnežđenja particija. **Partition** tag sadrži sledeća svojstva (atribute) - **Name**, **State**, **ImportLocation** i **Preserve**. **Name** svojstvo definiše naziv instance koja predstavlja particiju. U slučaju top level entiteta stavljamo naziv top level entiteta. U slučaju ostalih instanci navodimo kompletan hijerarhijski naziv instance počev od top level entiteta. **State** svojstvo ima dve moguće vrednosti *implement* i *import*. Vrednost *implement* označava da se particija u potpunosti rekompajlira. Vrednost *import* označava da se particija (tačnije struktura dela dizajna kojeg smo stavili u dotičnu particiju) uvozi iz rezultata prethodnog kompajliranja pre kreiranja particije. **Preserve** svojstvo definiše sa kog koraka kompajliranja se uvoze podaci za particiju. **Preserve** svojstvo ima četiri moguće vrednosti - *routing*, *placement*, *synthesis*, *inherit*. Vrednost *routing* označava da se uzima rezultat procesa rutiranja, vrednost *placement* da se uzima rezultat koraka postavljanja, a *synthesis* da se uzima rezultat koraka sinteze. U zavisnosti koji korak je uzet, svi koraci kompajliranja nakon dotičnog koraka se mogu ponovo izvršiti. Vrednost *inherit* označava da se nasleđuje vrednost od roditelja. Svojstvo **ImportLocation** označava odakle treba uzeti rezultat kompajliranja (sa koje lokacije), pri čemu ako je svojstvo **State** postavljeno na vrednost *implement*, onda se svojstvo **ImportLocation** postavlja na vrednost *NONE*.

Očigledno u datom primeru je navedeno da se sve particije u potpunosti rekompajliraju, pa će u izveštaju procesa analize i sinteze biti dat izveštaj o postojećim particijama (slika 3.3.15.1 i 3.3.15.2).

```
================================================
*                          Partition Report
================================================

Partition Implementation Status
-------------------------------

  Preserved Partitions:


  Implemented Partitions:

    Partition "/top_level_primer":
Attribute STATE set to IMPLEMENT.

    Partition "/top_level_primer/brojac_inst":
Attribute STATE set to IMPLEMENT.

    Partition "/top_level_primer/displej_inst":
Attribute STATE set to IMPLEMENT.

-------------------------------
```

Slika 3.3.15.1. – Deo izveštaja koji daje pregled postojećih particija

```
---------------------------
Partition Resource Summary:
---------------------------

Partition "/top_level_primer":
  Number of BUFG:                       1
  Number of bonded IOBs:               10

Partition "/top_level_primer/brojac_inst":
  Number of Slice Flip Flops:           4
  Number of Slice LUTs:                 4

Partition "/top_level_primer/displej_inst":
  Number of Slice LUTs:                 7

---------------------------
```

Slika 3.3.15.2. – Deo izveštaja koji daje pregled upotrebljenih resursa po participama

Na slici 3.3.15.1 je prikazan deo izveštaja procesa analize i sinteze u kome se navode sve detektovane particije u dizajnu. Na slici 3.3.15.2 je prikazan deo izveštaja procesa analize i sinteze u kome se navodi za svaku particiju dizajna koje resurse čipa su koristile. I ostali koraci kompajliranja daju pojedine detalje o participama, gde je najinteresantniji izveštaj koraka mapiranja, gde se dobija pregled realnih iskorišćenih resursa po participama. Napomenimo još da je u podešavanju projekta morao da bude promenjen čip na Virtex6 familiju da bi bila omogućena kreacija particija.

3.3.16. Programiranje FPGA čipa

Nakon što smo kreirali dizajn, dodelili pinove, kompajlirali dizajn i dobili konfiguracioni fajl za programiranje FPGA čipa, neophodno je i zaista programirati FPGA čip da bi on izvršavao dizajnirane funkcionalnosti. U okviru ovih skripti će ukratko biti prikazan metod programiranja FPGA čipa sa host računara. Kao primer će biti korištena Xilinx ML507 razvojna ploča koja sadrži FPGA čip Virtex5 XC5VFX70T-1FFG1136.

Tabela 3.3.16.1. – Raspored pinova po portovima dizajna

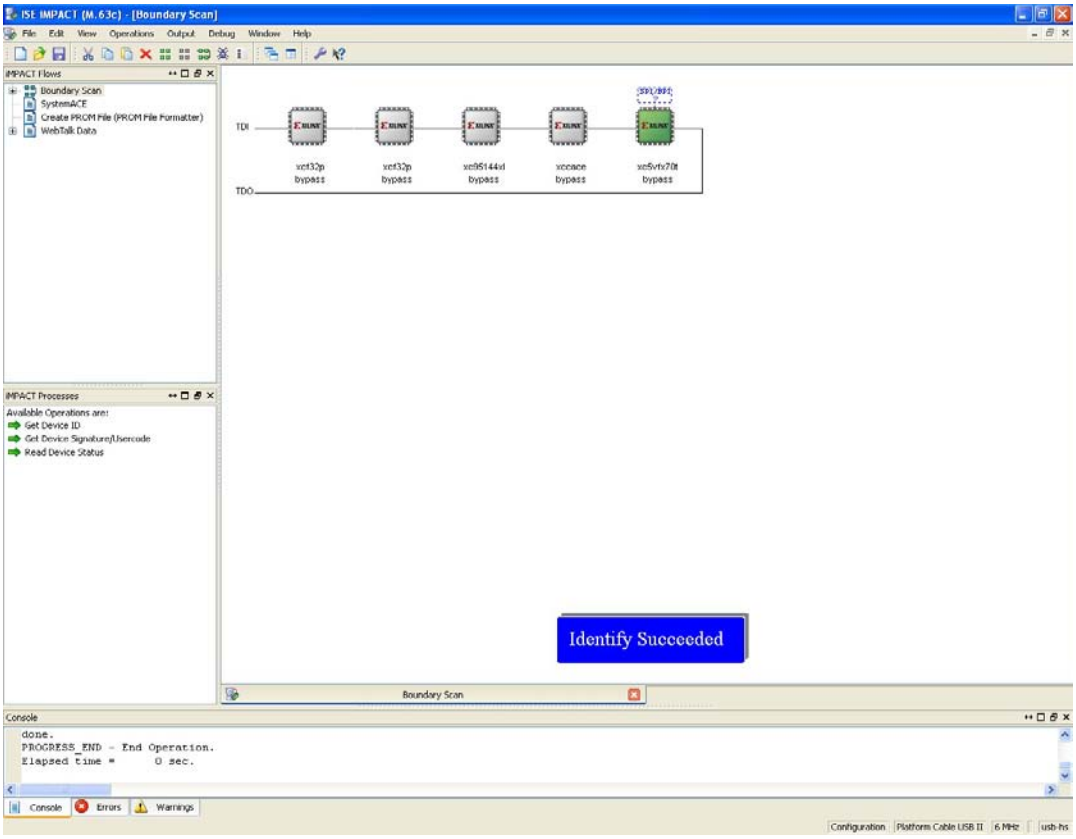
Port	Pin
clk	AH15
reset	U25
inc	AG27
displej_out(6)	H33
displej_out(5)	F34
displej_out(4)	H34
displej_out(3)	G33
displej_out(2)	G32
displej_out(1)	H32
displej_out(0)	J32

U okviru podešavanja projekta podesimo čip na Virtex5 XC5VFX70T-1FFG1136, i takođe izvršimo dodelu pinova u skladu sa tabelom 3.3.16.1. Kreirani .ucf fajl u kome je izvršena dodela pinova treba dodati u projekat (ako već nije dodat u projekat). U okviru .ucf fajla takođe treba definisati željenu frekvenciju takta. Sadržaj kreiranog .ucf fajla je:

```
NET "clk" LOC = "AH15";
NET "reset" LOC = "U25";
NET "inc" LOC = "AG27";
NET "displej_out(6)" LOC = "H33";
NET "displej_out(5)" LOC = "F34";
NET "displej_out(4)" LOC = "H34";
```

```
NET "displej_out(3)" LOC = "G33";
NET "displej_out(2)" LOC = "G32";
NET "displej_out(1)" LOC = "H32";
NET "displej_out(0)" LOC = "J32";
NET "clk" TNM_NET = clk;
TIMESPEC TS_clk = PERIOD "clk" 10 ns HIGH 50%;
```

Signal **clk** smo povezali sa 100MHz izvorom takta, signale **reset** i **inc** smo povezali sa prekidačima, dok smo **displej_out** povezali na pinove zaglavlja za proširenje (*expansion header*). Za razliku od Quartus primera nećemo modifikovati dizajn jer displej vodimo na zaglavlje za proširenje, a ne na sedmosegmentni displej jer na ML507 ploči ne postoji sedmosegmentni displej.



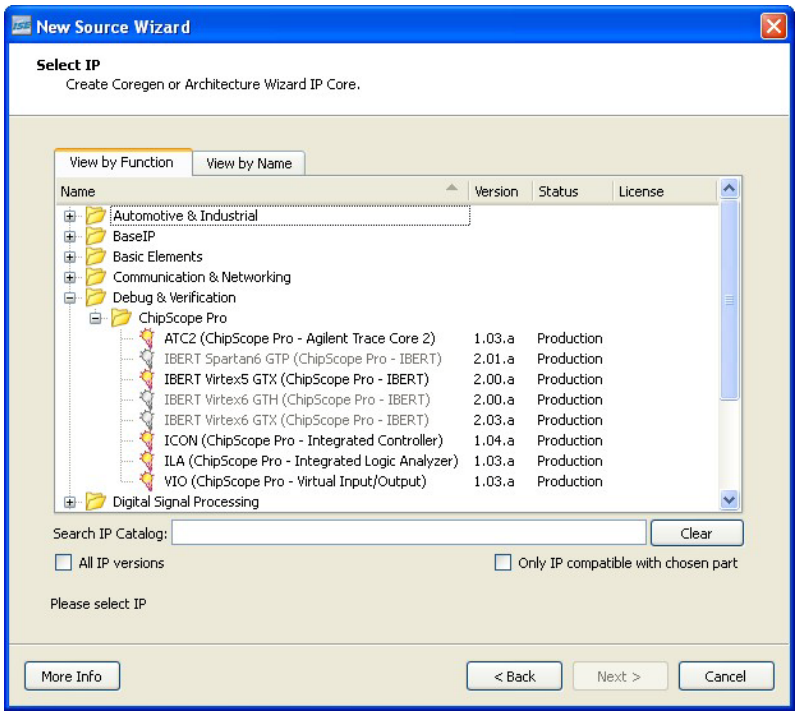
Slika 3.3.16.1. – Prozor za programiranje FPGA čipa sa host računara

Aktivirajmo kompletno kompajliranje dizajna. Kao rezultat procesa generisanja konfiguracionog fajla FPGA čipa dobijen je .bit fajl koji ćemo koristiti u procesu programiranja FPGA čipa sa host računara. Da bismo programirali FPGA čip na razvojnoj ploči neophodno je povezati ploču sa host računarom. U ovom slučaju se koristi običan USB kabl na koji je povezan Xilinxov hardver za programiranje preko JTAG interfejsa, koji je potom povezan na JTAG konektor na ploči. U zavisnosti od ploča mogu se koristiti i drugi tipovi kablova. Takođe, neophodno je povezati ploču na izvor napajanja i uključiti ploču. Tačan redosled operacija za povezivanje razvojne ploče sa host računarom (povezivanje ploče sa izvorom napajanja i sa host računarom, aktivacija ploče) treba izvršiti na osnovu dokumentacije same razvojne ploče. Nakon ovih operacija pokreće se programer iz ISE aplikacije. Programer se pokreće izborom opcije

Tools-> iMPACT iz glavnog menija čime se otvara prozor prikazan na slici 3.3.16.1. Treba aktivirati opciju **Boundary Scan** u koloni sa leve strane koja se koristi za skaniranje uređaja koji se mogu konfigurisati preko JTAG interfejsa. U glavnom delu prozora treba desnim klikom otvoriti meni i izabrati opciju **Initialize Chain** da bi se izvršilo skeniranje i dobila lista vidljivih uređaja (čipova) koji se mogu konfigurisati (slika 3.3.16.1). U donjoj koloni sa leve strane se nalazi lista akcija koje se mogu poduzeti za selektovani čip iz centralnog dela prozora. Nas interesuje da direktno programiramo FPGA čip sa host računara pa selektujemo FPGA čip kao što je prikazano na slici 3.3.16.1. Desnim klikom na FPGA čip otvara se meni u kome biramo opciju **Assign New Configuration File** da bismo dodelili željeni konfiguracioni fajl koji ćemo spustiti na FPGA čip (top_level_primer.bit). Izaberemo navedeni .bit fajl. Da bi izvršili programiranje neophodno je opet izvršiti desni klik na FPGA čip i izabrati opciju **Program** za aktivaciju procesa konfigurisanja FPGA čipa. Ista ova opcija se može izabrati i u donjoj levoj koloni nakon selekcije FPGA čipa u glavnom delu prozora (ova opcija se pojavljuje tek kad se izabere .bit fajl). U prozoru koji se otvara kliknemo na dugme OK čime se aktivira proces konfigurisanja (navedeni prozor se javlja samo kod prvog konfigurisanja novim .bit fajlom izabranim opcijom **Assign New Configuration File**, a posle u narednim programiranjima istim fajlom se ne pojavljaje navedeni prozor). Ako je konfigurisanje bilo uspešno, pojaviće se u dnu glavnog dela prozora poruka **Program Succeeded**.

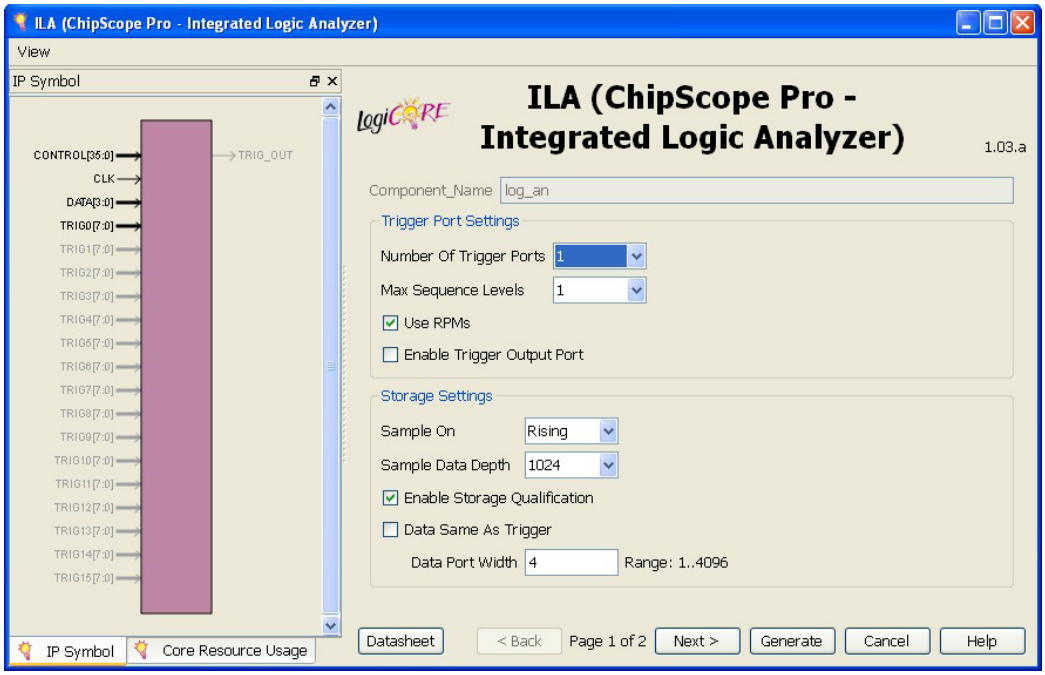
3.3.17. Interni logički analizator

ISE razvojno okruženje takođe nudi mogućnost internog logičkog analizatora kao i Quartus okruženje. U slučaju ISE okruženja interni logički analizator je dostupan upotrebom alata **ChipScope**, pri čemu je u ovom alatu integrisana i podrška za virtuelne ulaze i izlaze (sonde) kojima se stimulišu željeni događaji u dizajnu. U okviru ove sekcije koristićemo isti projekat i .ucf fajl kao i u prethodnoj sekciji.



Slika 3.3.17.1. – Grupa megafunkcija za ChipScope alat

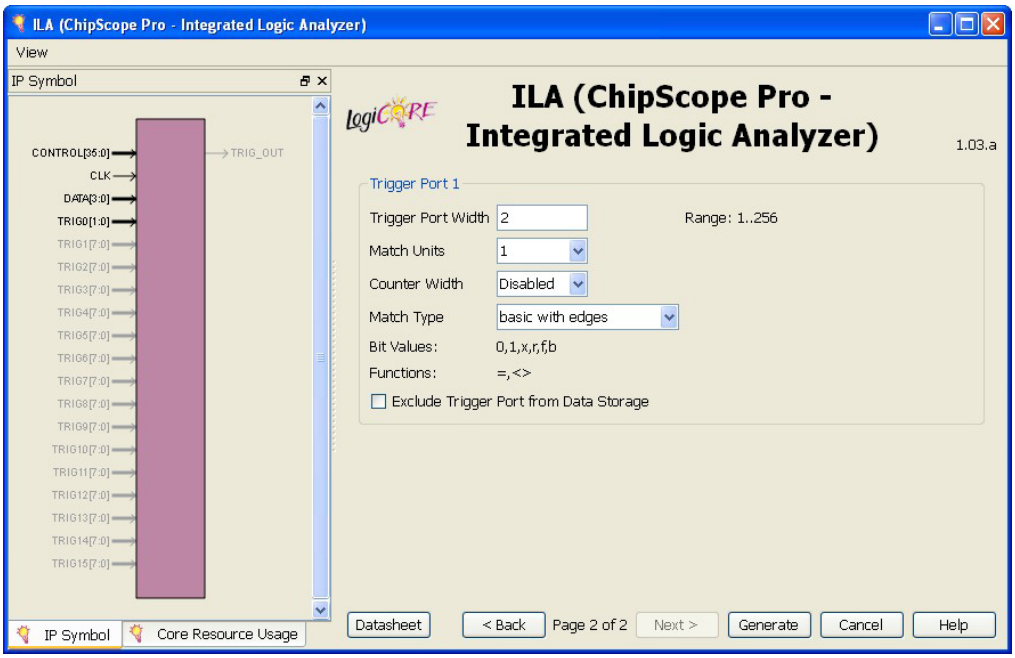
Da bi se omogućila upotreba internog logičkog analizatora neophodno je dodati **ChipScope** komponente u dizajn. Radi dodavanja internog logičkog analizatora i virtuelnih ulaza i izlaza, potrebno je kreirati komponentu kontrolera na koji se povezuju sve komponente logičkog analizatora i komponente virtuelnih ulaza i izlaza. Nazvaćemo ovu komponentu *ctrl*. Na nju ćemo povezati jedan logički analizator *log_an*, i jednu komponentu virtuelnog ulaza/izlaza *virt_io*. Sve ove komponente zahtevaju upotrebu megafunkcija koje su grupisane u **Debug & Verification->ChipScope Pro** grupi megafunkcija (slika 3.3.17.1). **ICON** megafunkcija je neophodna za komponentu kontrolera, **ILA** megafunkcija je neophodna za komponentu logičkog analizatora, a **VIO** megafunkcija za komponentu virtuelnog ulaza/izlaza. Navedene megafunkcije treba kreirati na neki od načina opisanih u sekciji koja je opisivala megafunkcije.



Slika 3.3.17.2. – Prvi prozor podešavanja ILA megafunkcije

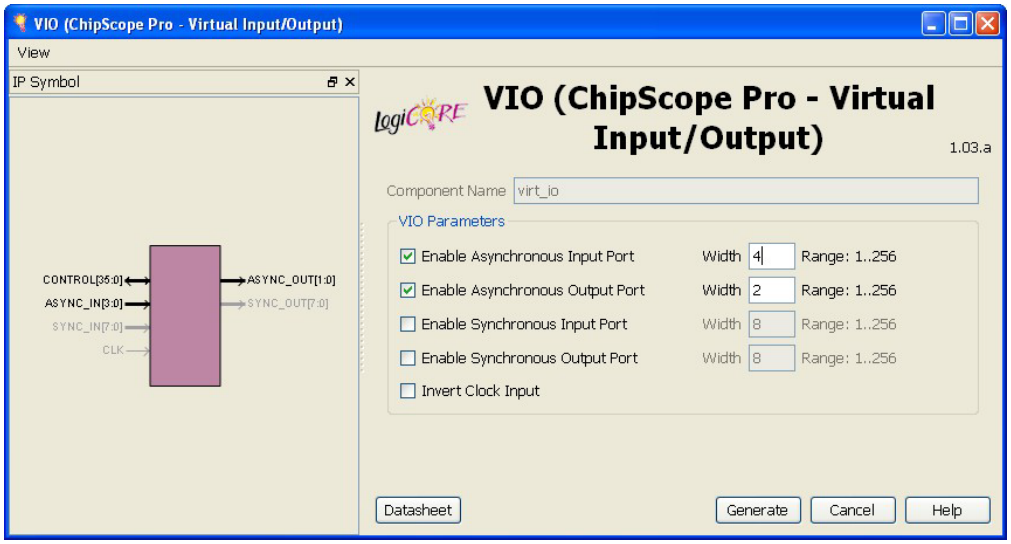
Prvi prozor u kreiranju **ILA** megafunkcije je prikazan na slici 3.3.17.2. U okviru ovog prozora se podešavaju *Trigger* i *Data* portovi. *Trigger* portovi igraju ulogu okidača na koji se izvršava akvizicija podataka, dok *Data* portovi igraju ulogu podataka koji se posmatraju. Može se podesiti opcija združivanja *Trigger* i *Data* signala čime posmatrani signali imaju osobine i *Trigger* i *Data* signala (**Data Same as Trigger**) i tada se koristi samo *Trigger* port. Kao što vidimo, imamo i podešavanja broja semplova koji će se uzeti prilikom akvizicije.

Drugi prozor podešavanja **ILA** megafunkcije je prikazan na slici 3.3.17.3. Na ovom prozoru se podešavaju mogućnosti *Trigger* signala, na primer mogu da se podese samo osnovna definisanja trigera (na primer okidanje na nivo signala) ili složenija. Takođe može da se aktivira i brojač *Trigger* događaja ako je bitno da se broji koliko puta se desio neki događaj. Može se aktivirati i opcija da se vrednosti *Trigger* signala ne uzimaju u procesu akvizicije. U dokumentaciji dostupnoj preko dugmeta **Datasheet** mogu se naći detaljnije informacije o podešavanjima **ILA** megafunkcije.



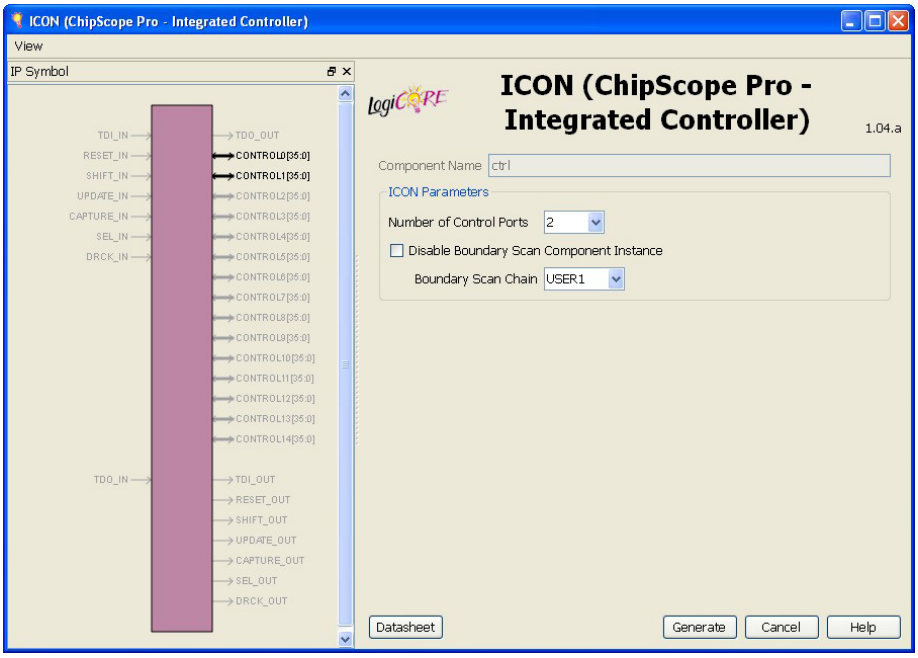
Slika 3.3.17.3. – Drugi prozor podešavanja ILA megafunkcije

Na slici 3.3.17.4 je prikazan jedini prozor za konfigurisanje **VIO** megafunkcije. U okviru ovog prozora se podešavaju virtuelni ulazi i izlazi (preciznije njihov broj), pri čemu ulazi i izlazi mogu biti asinhroni ili sinhroni.



Slika 3.3.17.4. – Prozor podešavanja VIO megafunkcije

Na slici 3.3.17.5 je prikazan jedini prozor za konfigurisanje **ICON** megafunkcije. U okviru ovog podešavanja je jedino bitno da se konfiguriše broj komponenti koje se povezuju na kontroler, odnosno broj kontrolnih portova **ICON** megafunkcije na koju će se povezati komponente **ILA** i **VIO** megafunkcija.



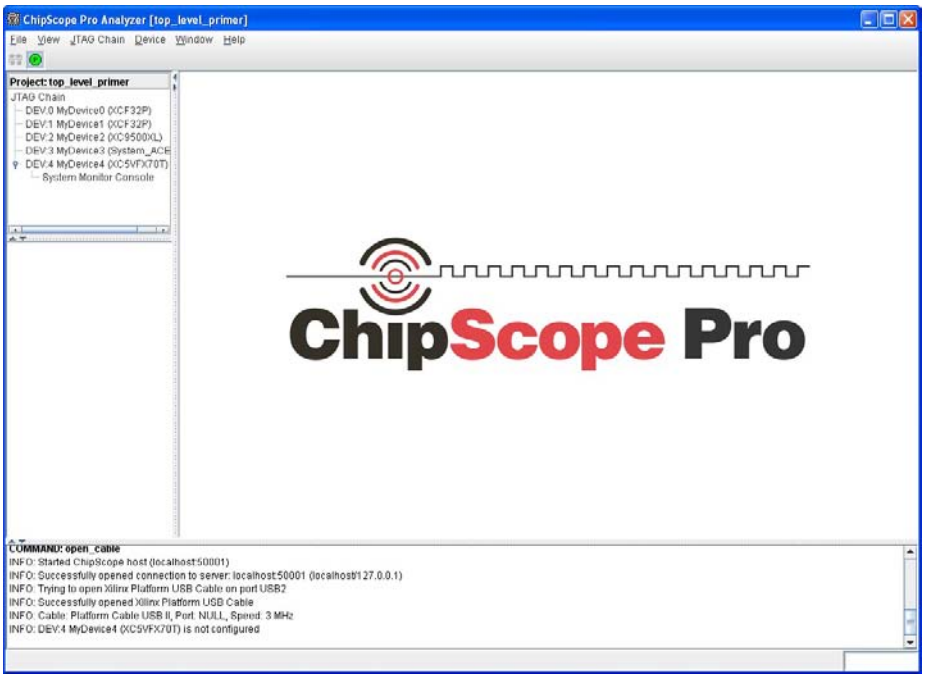
Slika 3.3.17.5. – Prozor podešavanja ICON megafunkcije

Nakon kreiranja navedenih megafunkcija, potrebno je ubaciti njihove komponente u dizajn. Ubacićemo ih u top level entitet čiji je VHDL kod sada:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;--neophodno jer radimo sa STD_LOGIC i STD_LOGIC_VECTOR
USE ieee.std_logic_unsigned.all;--neophodno za operaciju sabiranja

--top level entitet koji uvozi u vidu komponenti dekadni brojac i displej logiku
ENTITY top_level_primer IS
PORT
(
	clk:		IN STD_LOGIC;--signal takta
	reset:		IN STD_LOGIC;--asinhroni signal reseta
	inc:		IN STD_LOGIC;--kontrolni signal za dozvolu brojanja
	displej_out:	OUT STD_LOGIC_VECTOR(6 DOWNTO 0)--izlaz koji kontrolise displej
);
END top_level_primer;

ARCHITECTURE shema OF top_level_primer IS
	--deklaracija komponente dekadnog brojaca
	COMPONENT dekadni_brojac IS
	PORT
	(
		clk:	IN STD_LOGIC;--signal takta
		reset:	IN STD_LOGIC;--asinhroni signal reseta
		inc:	IN STD_LOGIC;--kontrolni signal za dozvolu brojanja
		q:	OUT STD_LOGIC_VECTOR(3 DOWNTO 0)--vrednost brojaca
	);
	END COMPONENT;
	--deklaracije logike za kontrolu displeja
	COMPONENT displej IS
	PORT
	(
		broj:	IN STD_LOGIC_VECTOR(3 DOWNTO 0);--vrednost broja na ulazu
		displej:	OUT STD_LOGIC_VECTOR(6 DOWNTO 0)--vektor ciji svaki indeks utice na jednu diodu displeja
```

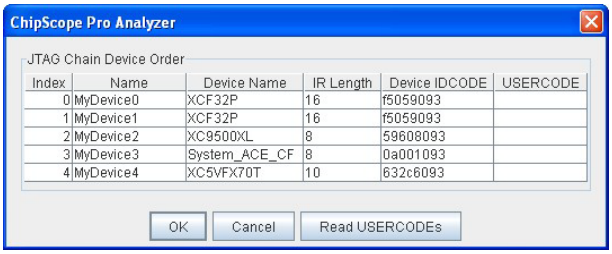
```
        );
        END COMPONENT;

        COMPONENT ctrl IS
        PORT
        (
                CONTROL0 : inout STD_LOGIC_VECTOR ( 35 downto 0 );
                CONTROL1 : inout STD_LOGIC_VECTOR ( 35 downto 0 )
        );
        END COMPONENT;

        COMPONENT log_an IS
        PORT
        (
                CLK : in STD_LOGIC := 'X';
                CONTROL : inout STD_LOGIC_VECTOR ( 35 downto 0 );
                TRIG0 : in STD_LOGIC_VECTOR ( 1 downto 0 );
                DATA : in STD_LOGIC_VECTOR ( 3 downto 0 )
         );
        END COMPONENT;

        COMPONENT virt_io IS
        PORT
        (
                CONTROL : inout STD_LOGIC_VECTOR ( 35 downto 0 );
                ASYNC_OUT : out STD_LOGIC_VECTOR ( 1 downto 0 );
                ASYNC_IN : in STD_LOGIC_VECTOR ( 3 downto 0 )
         );
        END COMPONENT;
        --deklaracija internog signala neophodnog za povezivanje izlaza dekadnog brojaca
        -- sa ulazom u displej logiku
        SIGNAL broj_int:STD_LOGIC_VECTOR(3 DOWNTO 0);
        SIGNAL ctrl0,ctrl1:STD_LOGIC_VECTOR(35 DOWNTO 0);
        SIGNAL trig0:STD_LOGIC_VECTOR(1 DOWNTO 0);
        SIGNAL host_input:STD_LOGIC_VECTOR(1 DOWNTO 0);
        SIGNAL reset1:STD_LOGIC;
        SIGNAL inc1:STD_LOGIC;
BEGIN
        --reset i inc se kontrolisu  prekidacima i sa host racunara
        reset1<=reset OR host_input(0);
        inc1<=inc OR host_input(1);
        --instanciranje komponente brojaca
        brojac_inst: dekadni_brojac
        PORT MAP
        (
                clk => clk,
                reset => reset1,
                inc => inc1,--vezujemo za interni signal
                q => broj_int
        );
        --instanciranje komponente za kontrolu displeja
        displej_inst: displej
        PORT MAP
        (
                broj => broj_int,
                displej => displej_out
        );
        trig0<=inc & reset;
        --instanca kontrolera za konekciju sa ChipScope alatom
        ctrl_inst: ctrl
```

```
PORT MAP
(
        CONTROL0 =>ctrl0,
        CONTROL1 =>ctrl1
);
--instanca internog logickog analizatora
 log_an_inst: log_an
PORT MAP
 (
        CLK =>clk,
        CONTROL =>ctrl0,
        TRIG0 =>trig0,--triger ce biti reset i inc
        DATA =>broj_int--posmatramo vrednost brojaca
 );
--instanca komponente virtuelnih ulaza i izlaza
virt_io_inst: virt_io
PORT MAP
 (
        CONTROL =>ctrl1,
        ASYNC_OUT =>host_input,--za kontrolu inc i reseta sa host racunara
        ASYNC_IN =>broj_int--posmatramo vrednost brojaca
);

END shema;
```



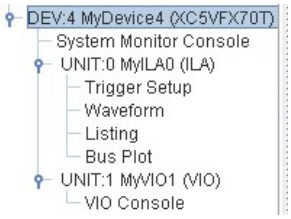
Slika 3.3.17.6. – ChipScope alat

Da bi aktivirali **ChipScope** alat, potrebno je aktivirati opciju **Analyze Design Using ChipScope** iz toka projekta čime se otvara prozor **ChipScope** alata prikazan na slici 3.3.17.6. Klikom na ikonicu s krajnje leve strane u redu ikonica ispod glavnog menija se pokreće skeniranje uređaja koji se vide preko JTAG interfejsa. Lista nađenih uređaja se prikazuje u posebnom prozoru (slika 3.3.17.7) i klikom na dugme **OK** oni se dodaju u gornju kolonu sa leve strane **ChipScope** prozora (slika 3.3.17.6).

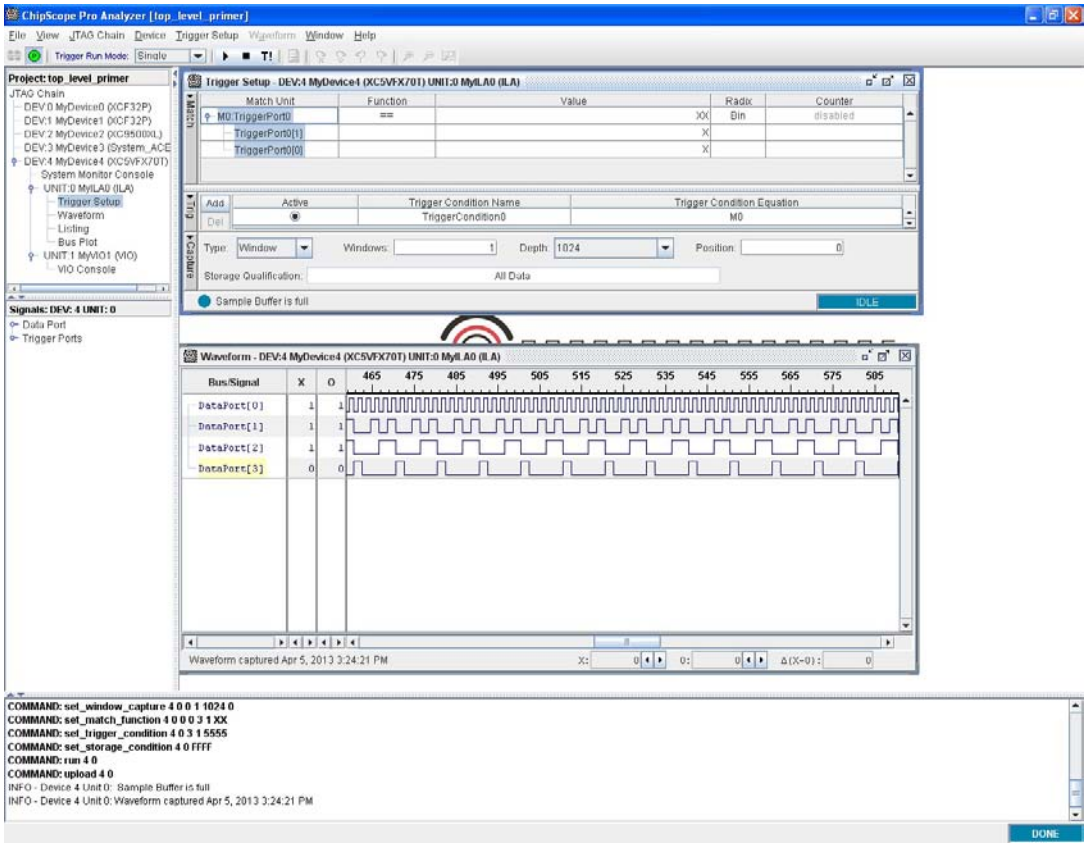


Slika 3.3.17.7. – Lista nađenih uređaja (čipova)

U listi uređaja selektujemo FPGA čip i desnim klikom na njega dobijamo meni u kome biramo opciju **Configure** za konfigurisanje (programiranje) FPGA čipa. U otvorenom prozoru se može selektovati željeni .bit fajl i potom se može pokrenuti proces konfigurisanja FPGA čipa. Kada je čip konfigurisan, tada se dodaju, ispod FPGA čipa u listi uređaja u gornjoj koloni sa leve strane, komponente koje smo definisali – logički analizator i virtuelni ulazi/izlazi kao što se vidi sa slike 3.3.17.8.

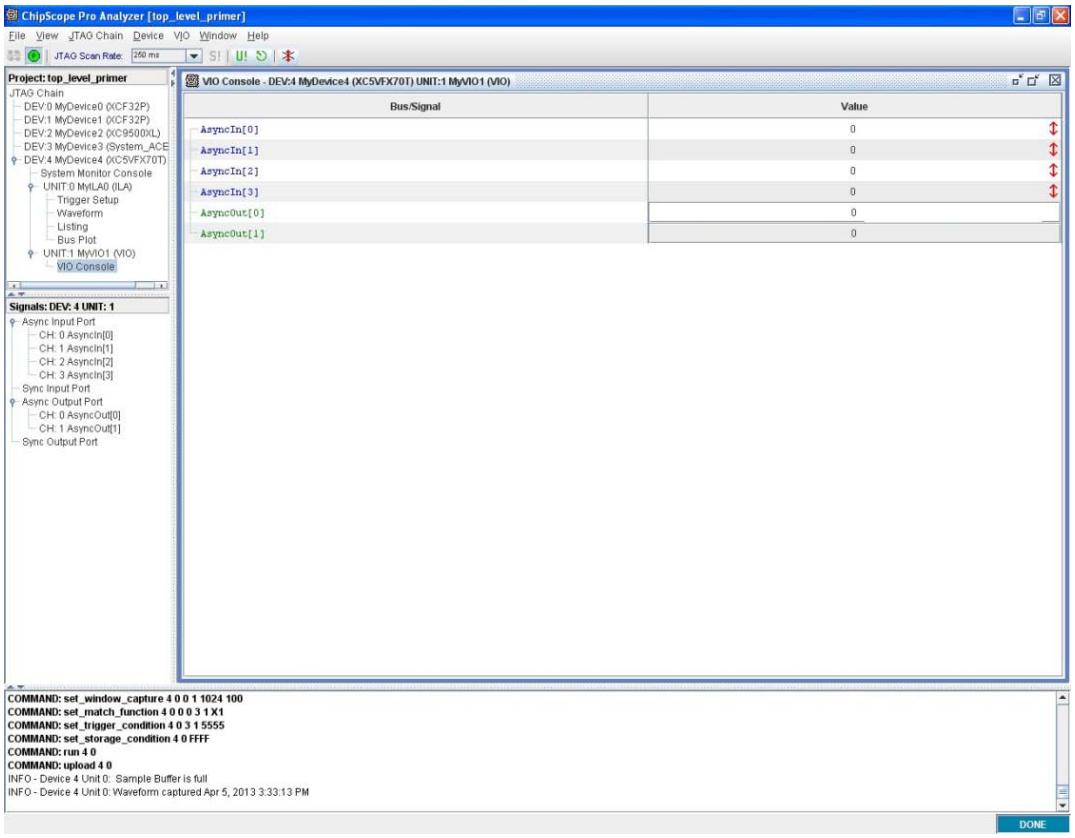


Slika 3.3.17.8. – Dodate komponente internog logičkog analizatora



Slika 3.3.17.9. – Prikaz praćenih signala i dela za podešavanje signala okidača

U slučaju prikaza logičkog analizatora, opcija **Trigger Setup** daje prozor za setovanje signala okidača (gornja polovina glavnog dela prozora – slika 3.3.17.9), dok **Waveform** daje prikaz izgleda praćenih signala (donja polovina glavnog dela prozora – slika 3.3.17.9). U **Trigger Setup** delu postoje tri podešavanja. **Match** podešavanje definiše vrednosti signala okidača na koju treba aktivirati akviziciju i koje se koriste u definisanju jednačine *triggera* u **Trig** podešavanju. **Trig** podešavanje nudi mogućnost definisanja više *trigger* uslova. Mogućnosti navedena dva podešavanja u najvećoj meri zavise od konfigurisanja **ILA** megafunkcije, odnosno kakve su mogućnosti za signale okidača omogućene (bazične ili proširene). **Capture** nudi podešavanje procesa akvizicije, broj uzetih odmeraka signala, kao i poziciju početka akvizicije. Broj uzetih odmeraka ne može biti veći od onoga koji je definisan u okviru konfigurisanja **ILA** megafunkcije. Pozicija početka akvizicije se definiše opcijom **Position**, gde vrednost u tom polju daje negativan ofset u odnosu na aktivaciju signala okidača (broj odmeraka koji je uzet pre aktivacije signala okidača). **Waveform** prikaz omogućava grupisanje signala u magistrale radi lakšeg posmatranja, a za magistrale nudi takođe izbor formata prikaza (binarni, heksadecimalni...). Donja kolona sa leve strane daje prikaz liste signala na **Data** i **Trigger** portu **ILA** komponente. Pokretanje akvizicije se vrši na dugme sa strelicom iz reda ikonica ispod glavnog menija, dok dugme kvadaratić označava nasilno zaustavljanje akvizicije. Dugme **T!** označava trenutnu aktivaciju akvizicije nezavisno od *trigger* uslova.



Slika 3.3.17.10. – Prikaz virtuelnih ulaza i izlaza

Na slici 3.3.17.10 je dat prikaz virtuelnih ulaza i izlaza kada je izabrana **VIO** komponenta i aktivirana dvostrukim klikom opcija **VIO Console**. Virtuelni ulazi predstavljaju sonde na kojima se posmatra promena signala koji se posmatra. Ova promena se u realnom vremenu prikazuje na ekranu monitora, pri čemu pošto su u pitanju asinhroni signali semplovanje se radi

na taktu JTAG interfejsa. Klikom na vrednost nekog virtuelnog izlaza možemo podesiti željenu vrednost čime možemo pokrenuti željene događaje u dizajnu koje želimo da analiziramo. Donja kolona sa leve strane daje prikaz liste virtuelnih ulaza i izlaza koje smo definisali u **VIO** komponenti.

Napomenimo da možemo istovremeno otvoriti i posmatrati navedene prozore za **Waveform** prikaz, za podešavanje signala okidača i uslova akvizicije, kao i prozor za kontrolu i posmatranje virtuelnih ulaza i izlaza. I ovde treba imati na umu da sve navedene komponente (**ICON**, **ILA** i **VIO**) troše resurse FPGA čipa i treba ih skloniti iz finalne testirane i verifikovane verzije dizajna. Takođe dodajmo da je poželjno da signal takta, koji se vezuje na **ILA** komponentu i na koju se sempluju podaci, bude bar dvostruko veće frekvencije od signala koji se posmatraju.