

**Uputstvo za rad sa Xilinx ISE 9.2i programskim
paketom – projektovanje i simulacija
korišćenjem VHDL-a**

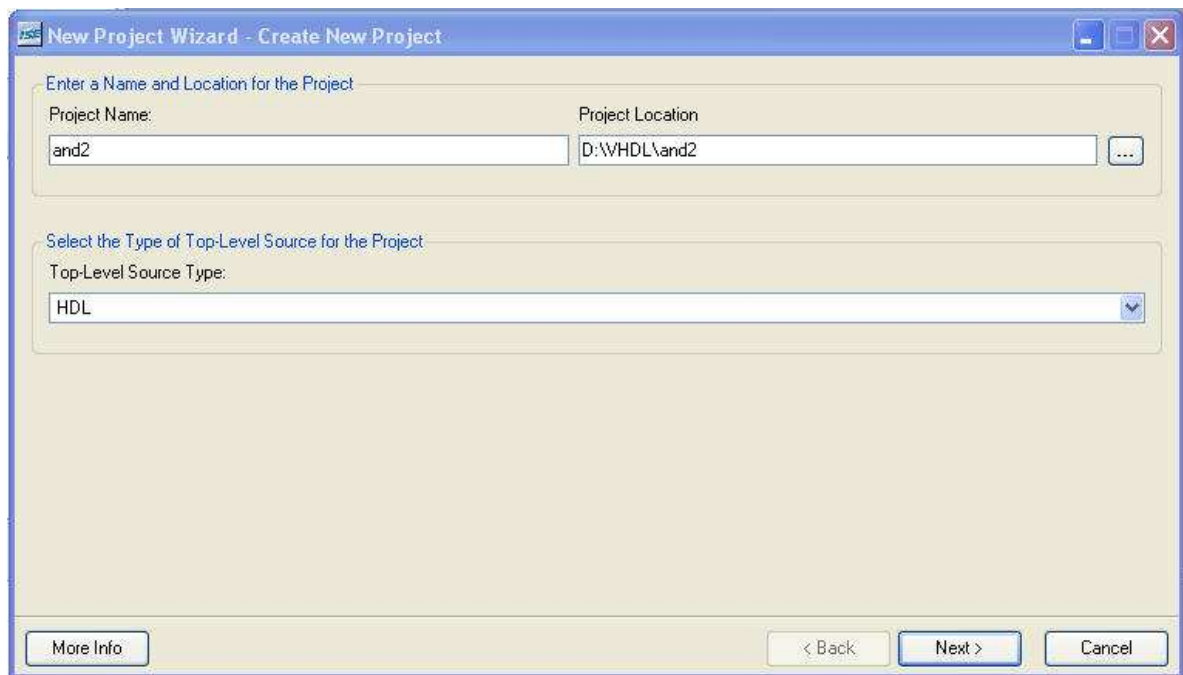
Novi Sad, novembar 2007.

Startovanje programskog paketa

Xilinx Foundation ISE 9.2i programski paket se aktivira dvoklikom na Project navigator ikonicu na desktopu ili aktiviranjem u Start meniju Windowsa iste opcije.

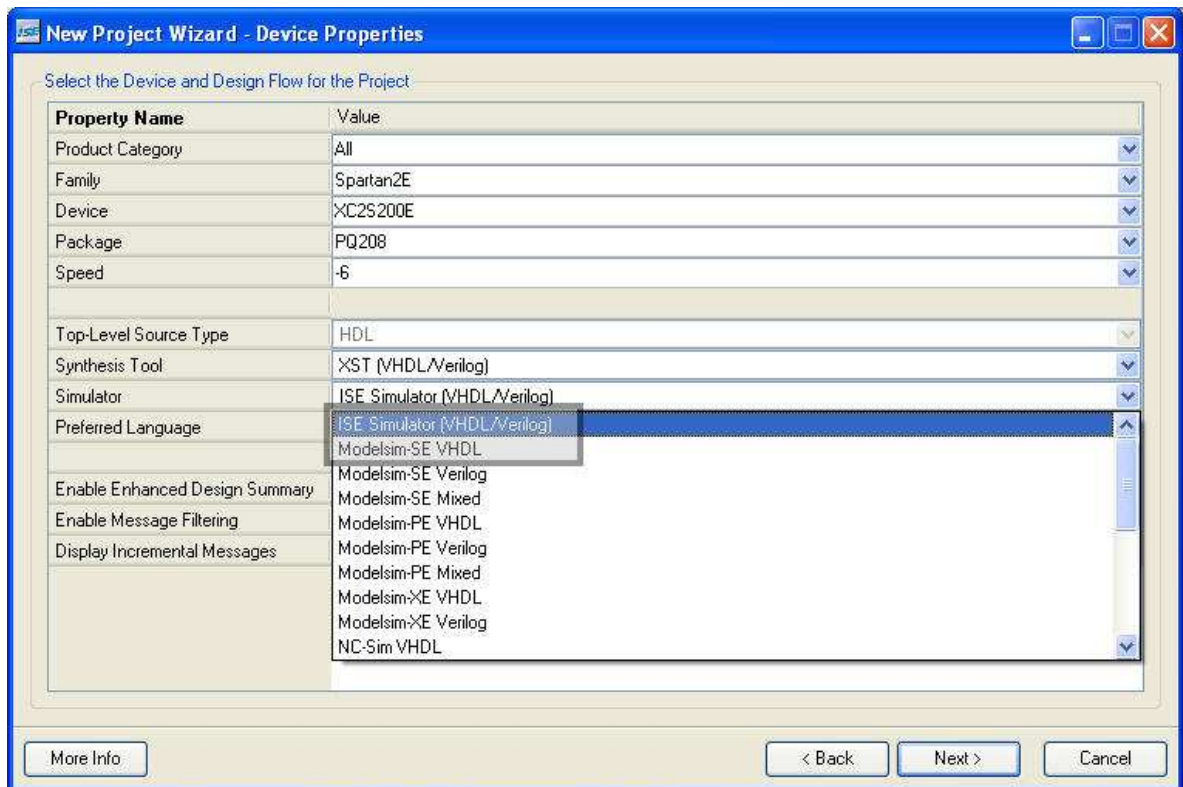
Kreiranje novog projekta

Kreiranje novog projekta počinje nakon odabira opcije *File* pa *New project* u meniju. Otvara se prozor (slika 1) u kojem treba popuniti naziv novog projekta, lokaciju gde će biti snimljen na disku i tip glavnog modula (mi ćemo raditi samo HDL tip).



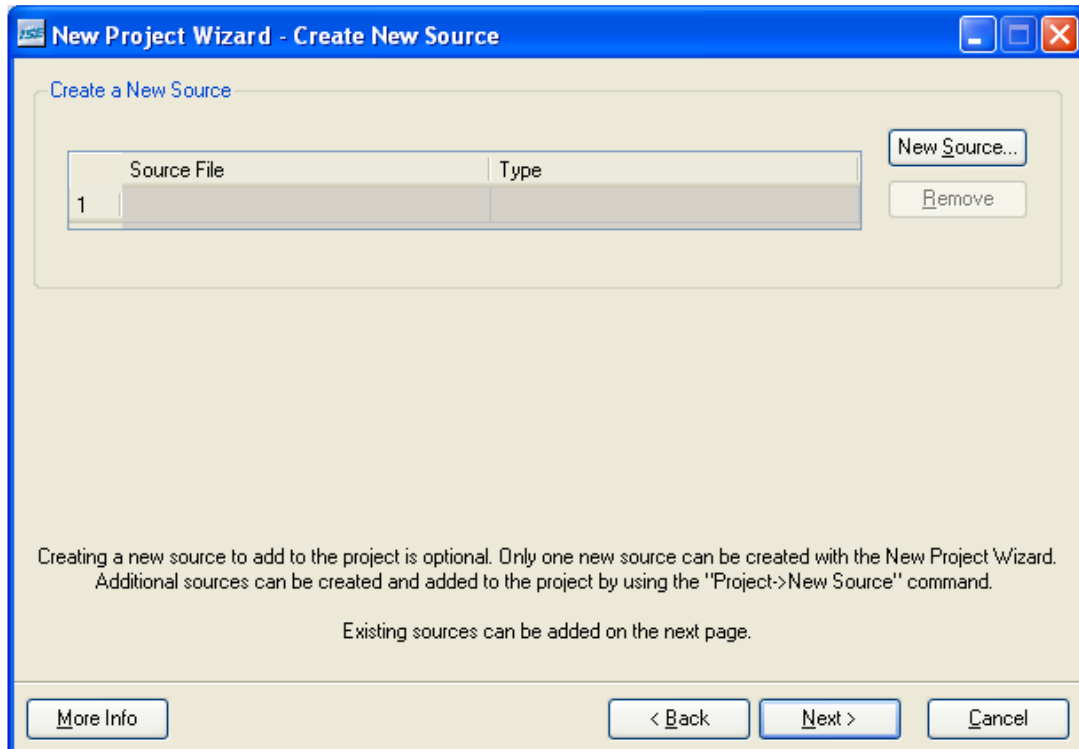
Slika 1: Kreiranje novog projekta

Nakon što smo uneli potrebne podatke (npr. Primer1), pritiskom na *Next* se otvara novi prozor (slika 2) u kome treba odabrati podatke o Xilinx komponenti koja će se koristiti. Treba podesiti familiju (Spartan2), naziv komponente (xc2s200e), kućište (pq208) i brzinu (-6). Pored toga treba odabrati i softverske alate. Mi ćemo koristiti Xilinxov softver za sintezu i za simulaciju (moguće je korišćenje drugih alata za simulaciju, npr Mentor Graphics-ov ModelSim). Sve će biti pisano u VHDLu



Slika 2: Podešavanje parametara

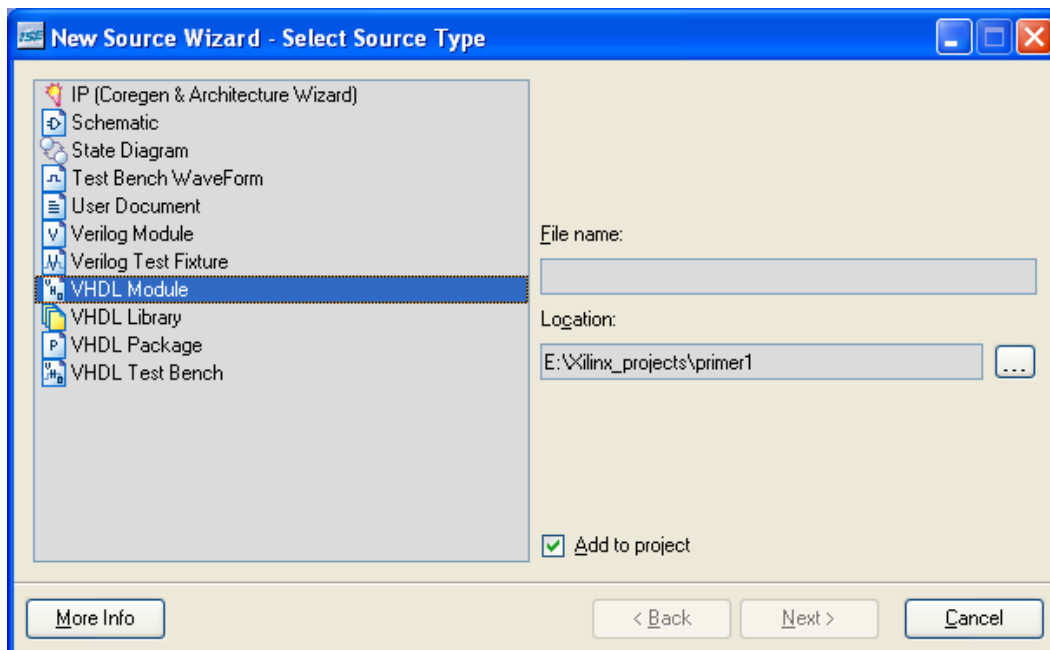
Nakon pritiska na taster *Next* pojavljuje se prozor (slika 3) koji omogućava kreiranje novog izvornog fajla.



Slika 3: Kreiranje izvornog fajla

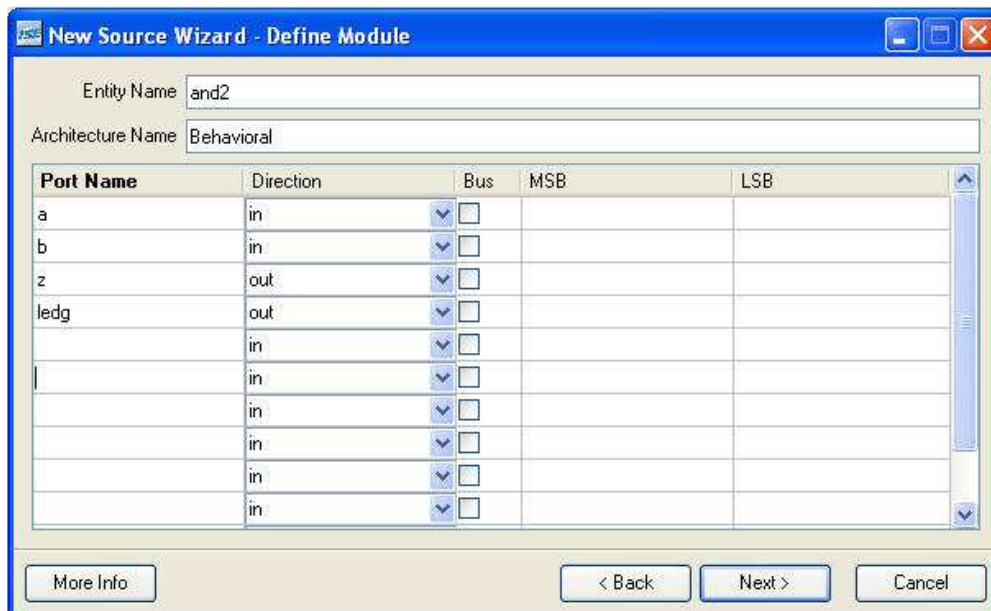
Pritiskom na *New Source* se otvara prozor (slika 4) za definisanje vrste novog izvornog programa, kao i dodelu imena. Recimo da ćemo ga nazvati isto primer1 kao i folder u

kojem će biti snimljeni svi fajlovi u vezi sa ovim projektom. Za tip izvornog programa treba odabrati *VHDL Module* opciju.



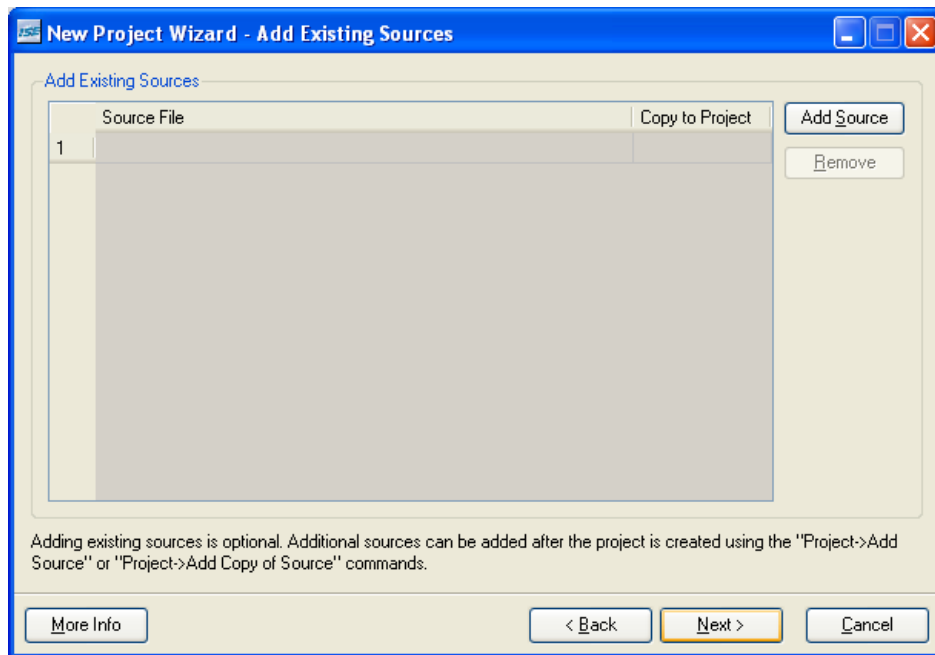
Slika 4: Izbor tipa izvornog programa

Nakon pritiska na *Next* otvara se prozor (slika 5) gde definišemo interfejs (portove) za novi izvorni VHDL program. U tabeli se navode imena portova i tip. Portovi mogu biti ulazni (*in*), izlazni (*out*) ili ulazno-izlazni (*inout*). Ukoliko je neki signal višebitni, onda se u ovom prozoru navode i njegove dimenzije.



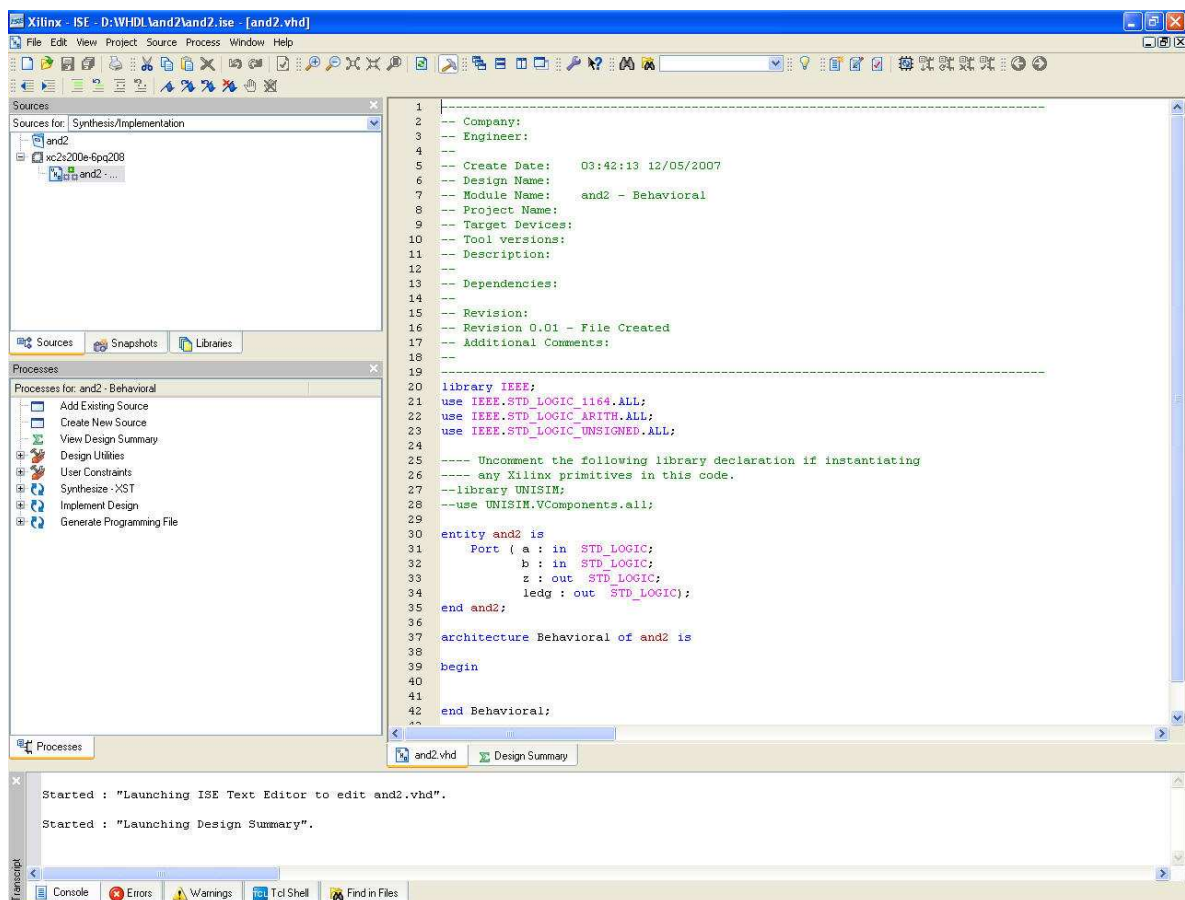
Slika 5: Definisanje interfejsa

Nakon pritiska na *Next* se pojavljuje prozor sa informacijama o novom VHDL programu. Pritiskom na *Finish* se kreira novi VHDL izvorni fajl koji se pojavljuje u prozoru odakle smo počeli kreiranje fajla (kao na slici 4). Pritiskom na *Next* se otvara novi prozor (slika 6) u kome možemo da dodamo neke već postojeće izvorne programe.



Slika 6: Dodavanje postojećih izvornih programa

Potom pritiskom na *Next* dobijamo prozor sa informacijama o novom projektu. Konačno, pritiskom na *Finish* smo završili postupak kreiranja novog projekta (slika 7).



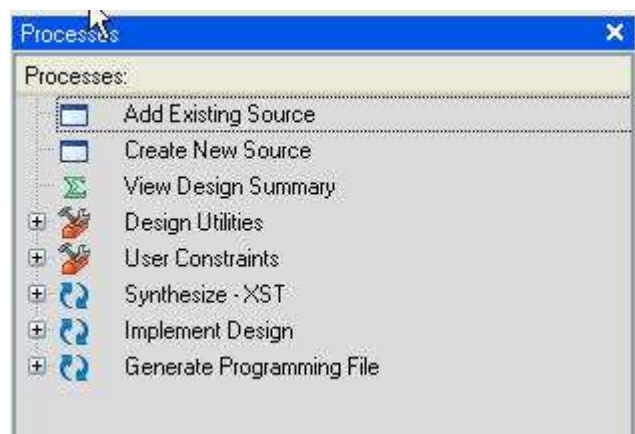
Slika 7: Izgled novog projekta

Pre bilo kakvog daljeg rada treba dopuniti VHDL program tako da radi nešto konkretno. Kao najjednostavniji primer uzećemo da su a i b ulazi dvoulaznog I kola, a z izlaz. Dopunjeni kod izgleda kao na slici 8.

```
architecture Behavioral of primer1 is
begin
    z <= a and b;
    ledg <= '1';
end Behavioral;
```

Slika 8: Dopunjen kod

U prozoru u kojem se aktiviraju procesi (slika 9) odabrati *Synthesize – XST*, pa opciju *View RTL Schematic* koja vrši sintaksnu proveru koda i kreira elektronsku šemu na osnovu zadatog koda. Ova opcija je veoma zgodna da se proverí da li je Xilinx softver napravio hardver onako kako smo zamislili ili su potrebne korekcije.



Slika 9: Procesi u projektovanju

Simulacija

Da bismo pojednostavili objašnjenje simulacije u VHDLu zamislićemo situaciju realnog merenja nekog digitalnog sistema. Ovakav merni sistem bi se sastojao od:

- 1) uređaja koji testiramo
- 2) generatora ulaznih signala
- 3) mernih instrumenata kojima merimo i posmatramo talasne oblike izlaznih signala

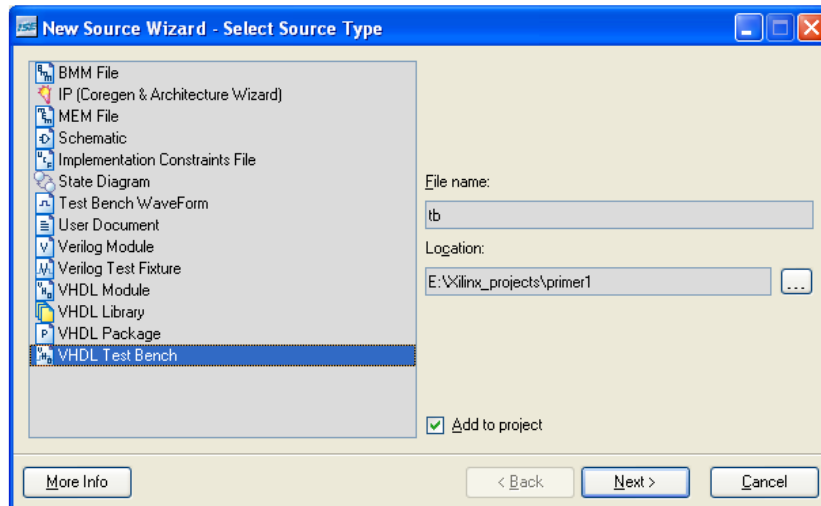
U VHDLu imamo potpunu analogiju sa gore pomenutim mernim sistemom. Uređaj koji testiramo bi bio VHDL program koji opisuje funkcionisanje digitalnog sistema koji simuliramo. Generator ulaznih signala se realizuje u posebnom VHDL programu tako da se definišu i generišu svi potrebni ulazni signali, a merni instrument nam je simulator u kome posmatramo talasne oblike izlaznih signala.

Kako generisati test signale a i b u VHDL test programu? Prisjetimo se da svaki VHDL program, osim dela gde se uključuju biblioteke, ima još dva dela – deo gde se definiše entitet i deo gde se definiše arhitektura. I VHDL test programi imaju ovakvu strukturu. Prva razlika je u tome što se u delu za definisanje entiteta ne navode portovi, ali se zato u arhitekturnom delu uključuje kao komponenta (sa pripadajućim portovima) entitet koji simuliramo. Potom se definišu signali čije ćemo talasne oblike posmatrati. U telu

arhitekturnog dela se potom ti signali povezuju sa portovima komponente. Konačno se definišu i svi potrebni ulazni signali.

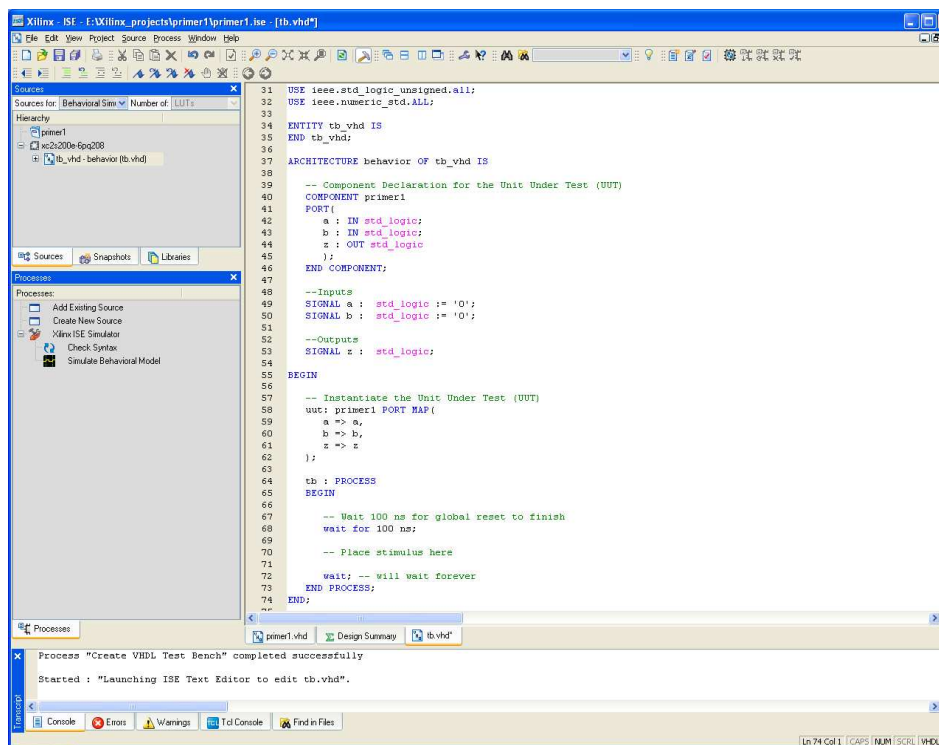
Kako sve to uraditi u Xilinx Foundation ISE-u?

Ukoliko želimo da radimo simulaciju, potrebno je dodati u projekat VHDL *test bench* program (kraće *tb*). To je program u kome ćemo zadati ulazne test signale – test vektore. Da bi smo dodali *tb* program, treba odabrati opciju *Project* iz menija i potom *New Source*. Ovim dobijamo prozor koji smo već videli na slici 4, ali ćemo ovog puta za tip fajla odabrati *VHDL Test Bench* (slika 10) i daćemo ime programu *tb*.



Slika 10: Dodavanje *test bench* fajla

Pritiskom na *Next* se otvara prozor u kome se odabira koji je izvorni program koji simuliramo. S obzirom da je ovo trivijalan projekat odaberemo *primer1*. Nakon *Next*, slede informacije o kreiranom fajlu i posle *Finish* smo dodali test bench fajl u projekat. Izgled prozora nakon ove operacije je dat na slici 11.



Slika 11: Glavni prozor nakon dodavanja *test bench* programa

Xilinx ISE nam daje kostur fajla (prikazan u uokvirenom tekstu) za simulaciju sa definisanim dizajnom (entitetom) koji želimo da testiramo. Na nama je da smislimo efikasan način da proverimo funkcionalnost dizajniranog sistema.

```
ENTITY primer_tb IS
END primer_tb;

ARCHITECTURE behavior OF primer_tb IS
    COMPONENT primer
    PORT(
        A, B : IN std_logic;
        Z : OUT std_logic
    );
    END COMPONENT;

    SIGNAL A, B, Z : std_logic;

BEGIN

    uut: primer PORT MAP(
        A => A,
        B => B,
        Z => Z
    );

    -- *** Test Bench - User Defined Section ***
    tb : PROCESS
    BEGIN
        wait; -- will wait forever
    END PROCESS;
    -- *** End Test Bench - User Defined Section ***

END;
```

Program ćemo dopuniti sa test vektorima koji omogućavaju sve četiri kombinacije na ulazima i kola koje testiramo:

```
tb : PROCESS
BEGIN

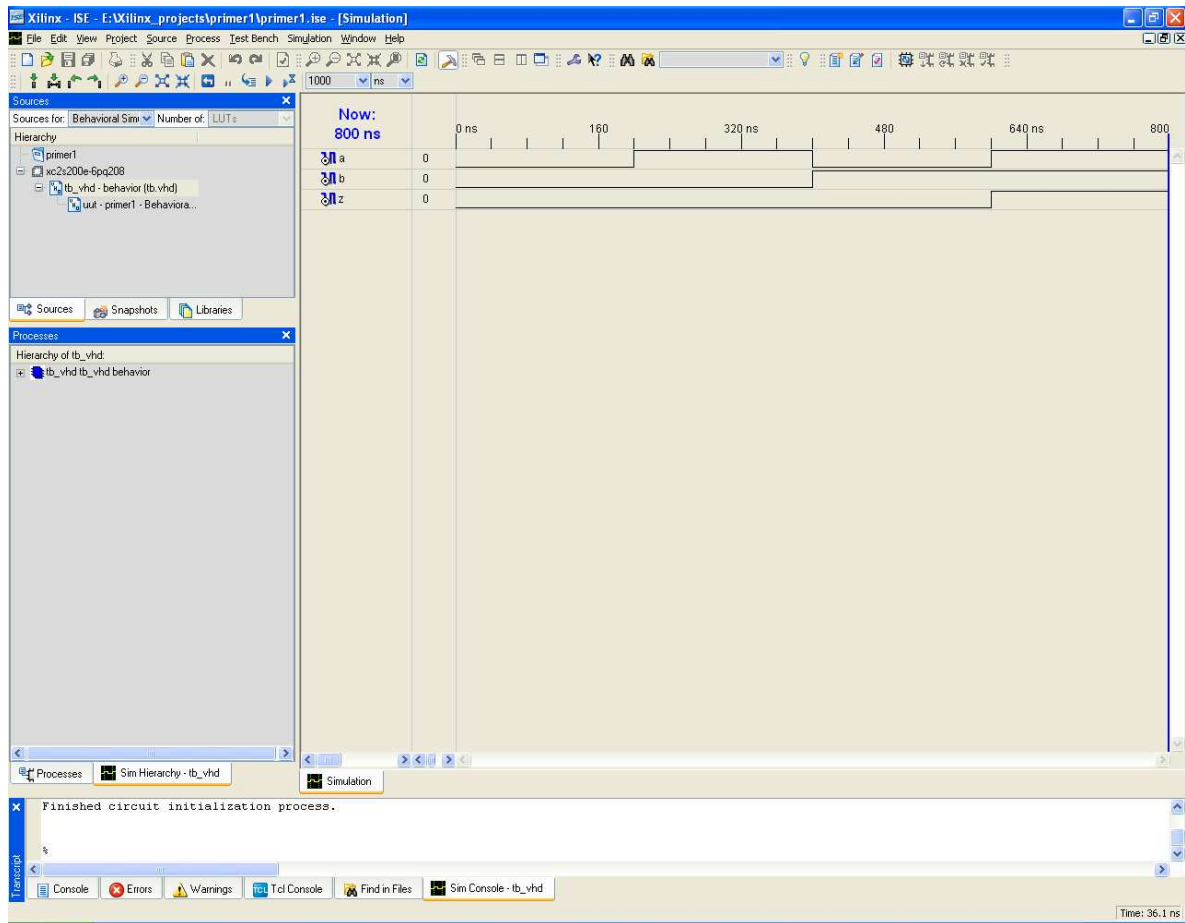
    a <= '0', '1' after 200 ns, '0' after 400 ns, '1'
after 600 ns, '0' after 800 ns;
    b <= '0', '1' after 400 ns, '0' after 800 ns;

    wait; -- will wait forever
END PROCESS;
```

Slika 12: Dopunjeni *test bench* kod

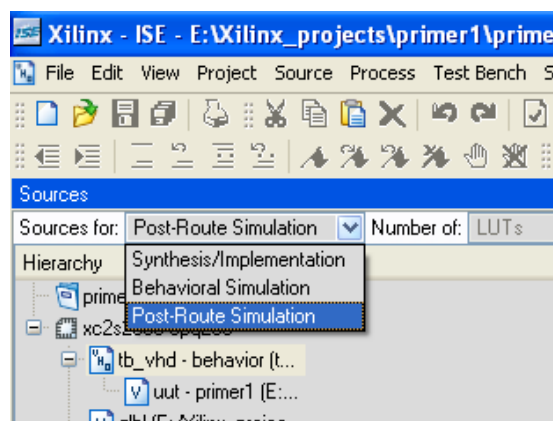
i nakon toga odabirom *Simulate Behavioral Model* možemo da počnemo simulaciju.

Nakon aktiviranja, skoro sva podešavanja urađena su automatski. Jedino po potrebi treba promeniti vreme trajanja simulacije (ono je postavljeno na 1000 ns). Rezultate simulacije vidimo na slici 13.

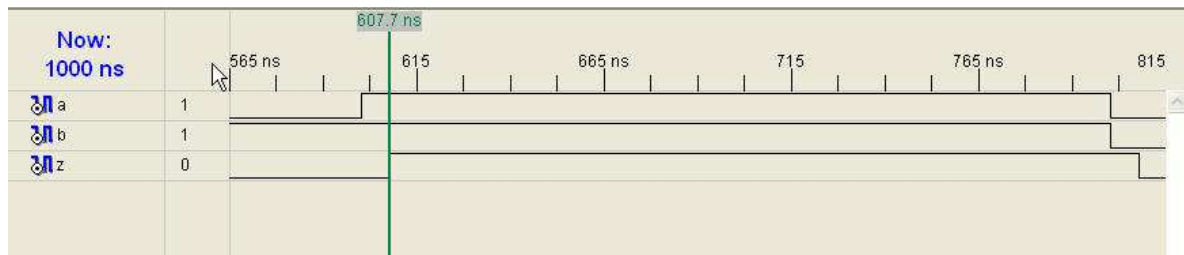


Slika 13: Rezultati simulacije

Ovim smo dobili rezultate osnovne funkcionalne ili tzv. *bihevijalne* simulacije. Ukoliko želimo da dobijemo preciznije rezultate onda ćemo raditi *Post Route* simulaciju. Odabir sa kojim izvornim fajlovima radimo se vrši u combo box-u prikazanom na slici 14. Ona se aktivira isto kao i prethodna simulacija (s tim da je opcija *Simulate Post Place and Route Model*). Uvećani rezultat na slici 15 nam pokazuje da je kašnjenje izlaza za ulazima oko 7 ns što se u osnovnoj simulaciji ne vidi.



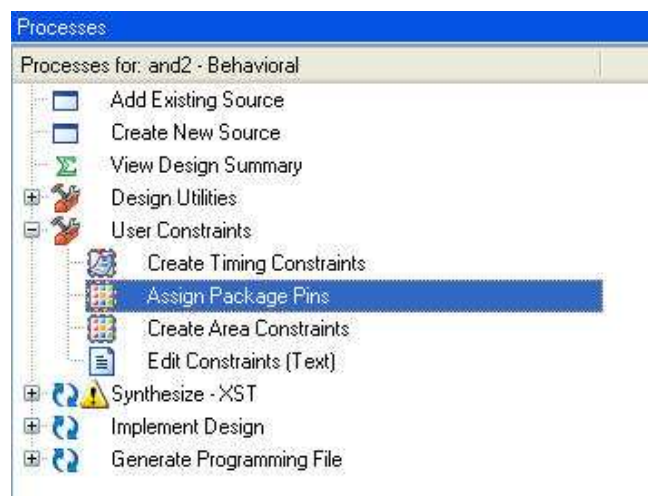
Slika 14: Izbor *Post-Route* simulacije



Slika 15: Uvećan rezultat *post-route* simulacije

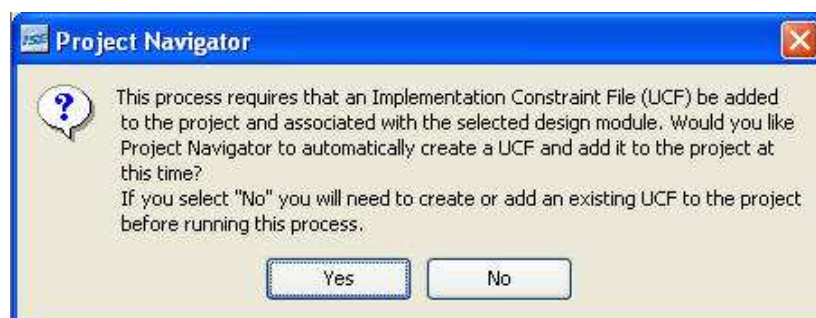
Povezivanje dizajna sa pinovima na FPGA kolu

Da bismo mogli da proverimo da li dizajnirani sistem zaista radi na FPGA kolu (pored simulacije) potrebno je da odgovarajuće portove našeg dizajna povežemo sa pinovima na FPGA kolu. Taj proces započinjemo pozivom User Constraints Editora, dvostrukim klikom na natpis Assign Package Pins



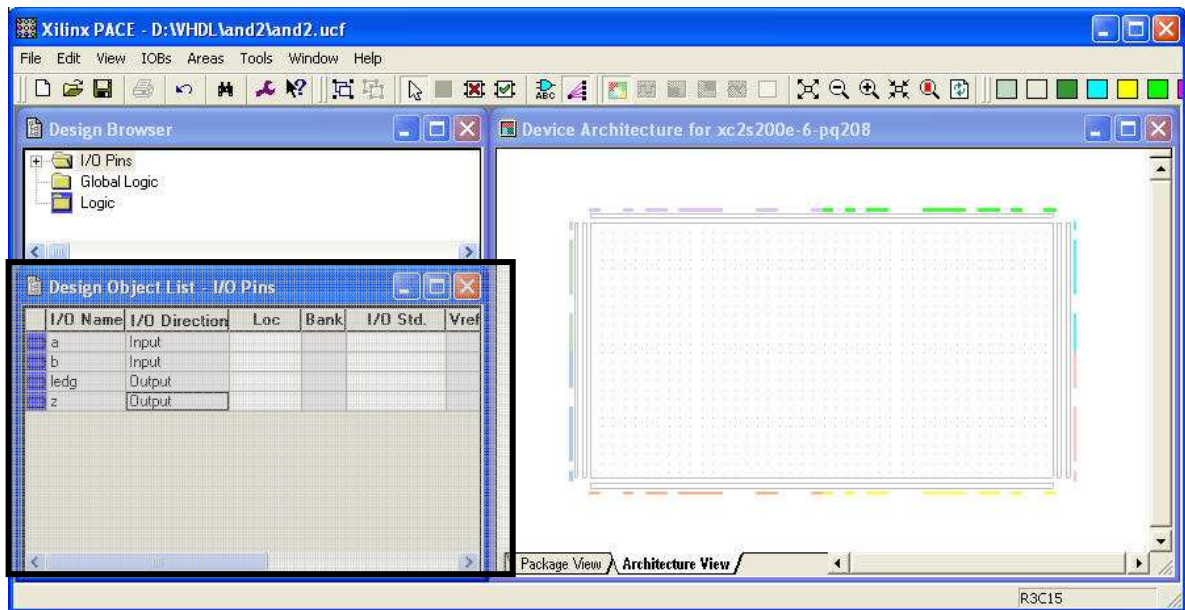
Slika 16: Proces kojim se započinje dodela pinova dizajnu

Xilinx ISE nas tada obaveštava da će pinove FPGA kola koje budemo dodelili našem dizajnu smestiti u poseban fajl koji bi trebalo da dodamo odgovarajućem modulu u dizajnu. Pitanje koje nam postavlja je da li želimo da on to automatski doda projektu nakon kreiranja.



Slika 17: Da li želimo da se neki procesi automatski obave?

Potom se otvara program Xilinx PACE programa kojim se vrši dodela pinova FPGA kola.



Slika 18: Xilinx PACE

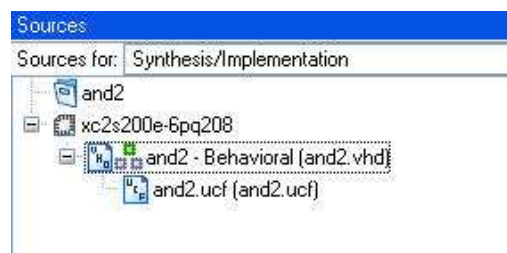
Deo koji je nama od interesa je *Design Object List – I/O Pins* (označen kvadratom na slici 18). U ovom delu vidimo portove našeg dizajna koje je potrebno povezati sa odgovarajućim pinovima. Povezivanje se vrši tako što u koloni *Loc* kod odgovarajućeg porta upišemo broj pina. Popunjena tabela je prikazana slici 19.

I/O Name	I/O Direction	Loc	Bank	I/O Std.	Vref
a	Input	p71	BANK		
b	Input	p69	BANK		
ledg	Output	p88	BANK		
z	Output	p111	BANK		

Slika 19: Pinovi dodeljeni dizajnu

Tabela na slici 19 je popunjena korišćenjem tabele u prilogu koja nam govori kojem pinu FPGA kola odgovara koja periferija na kartici DIO4 ako se kartica priključi na A1 – A2 konektore. Konkretno u ovom slučaju imamo dvoulazno I kolo koje je na ulazu povezano sa dva prekidača (SW1 i SW2), a na izlazu sa LED1, LEDG signal dozvoljava rad svetlećih dioda i on mora biti na logičkoj 1 da bi diode radile (zbog toga je i dodat u dizajn).

Ukoliko je proces dodele pinova uspešno završen odgovarajućem modulu (and2) će biti dodeljen odgovarajući fajl (kako je to prikazano na slici 20)



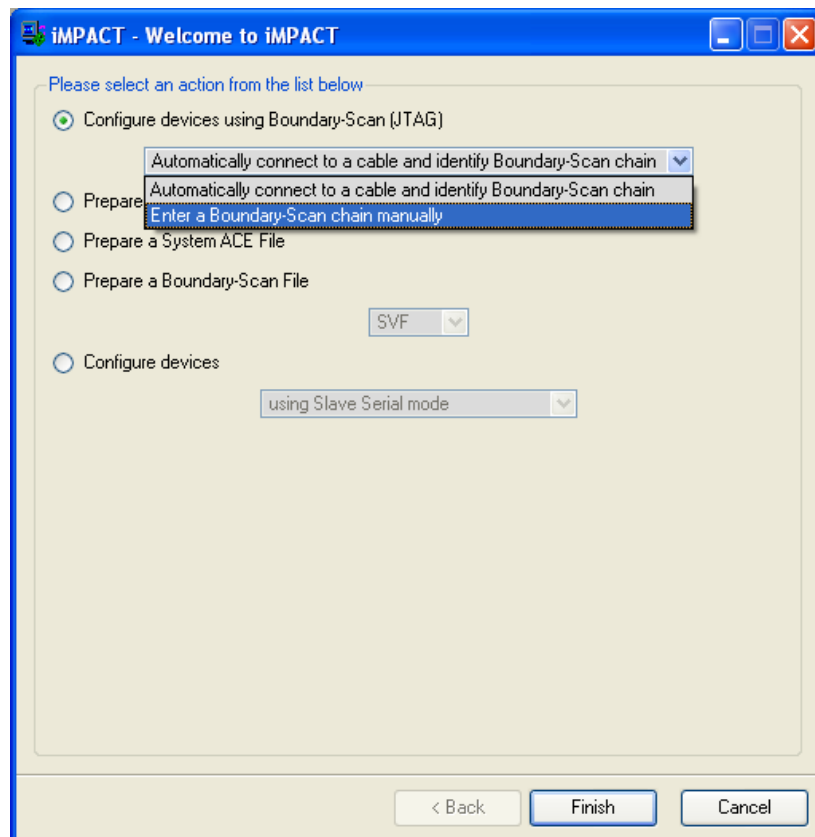
Slika 20: Proces dodele pinova dizajnu uspešan

Programiranje FPGA

Nakon uspešno urađene simulacije i implementacije, poslednji korak u projektovanju je implementacija dizajna na realnom hardveru. U našem slučaju, reč je o Xilinx razvojnom sistemu (Digilab).

Da bismo mogli da programiramo FPGA čipa na ovom razvojnom sistemu, potrebno je aktivirati proces koji to radi. U prozoru gde se pokreću procesi otvoriti *Generate Programming File* (videti sliku 9) pa potom *Configure Device* koji će aktivirati deo programskom paketa koji je zadužen za programiranje FPGA čipa. U prvom prozoru koji se otvori (slika 21) odabrati *Enter a Boundary-Scan chain manually* i pritisnuti *Finish*. U prozoru desnim klikom odabrati opciju *Add Xilinx Device* nakon koje se otvara fajl dijalog prozor gde bi trebalo da se nalazi fajl sa ekstenzijom *.bit* i imenom pod kojim je i projekat snimljen. Otvoriti taj fajl dvoklikom ili selekcijom uz pritisak na *Open*. (slika 22)

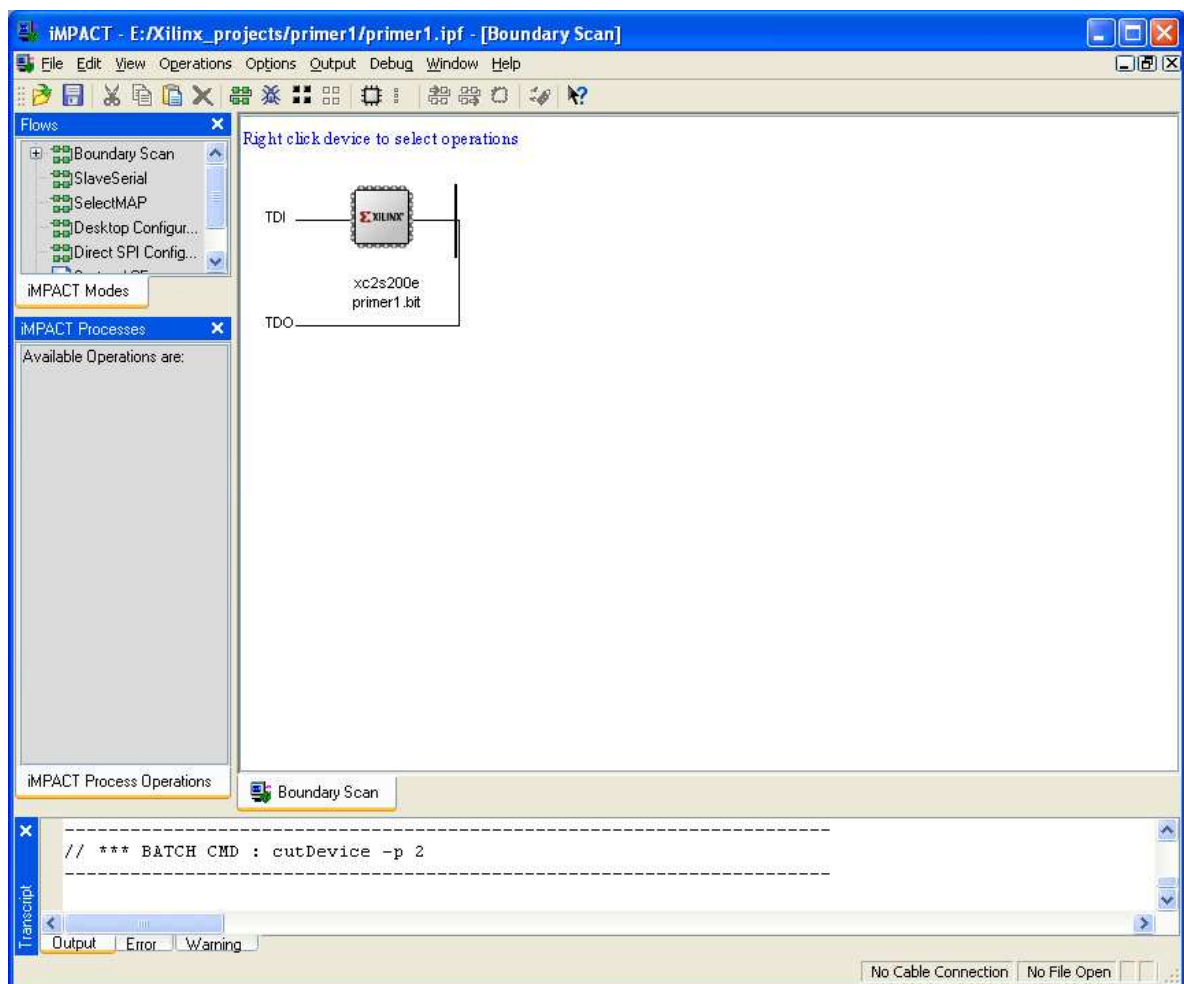
Upozorenje koje se pojavi zatvoriti pritiskom na *OK*. Nakon ovoga je urađena inicijalizacija Boundary-Scan Moda i sada desnim klikom na kvadrat koji simboliše FPGA čip (slika 23) odabrati Program ociju iz pop-up menija. Pritiskom na *OK*, počinje proces prvo povezivanja programa sa programatorom, a potom i samo programiranje čipa. Uspešan proces programiranja je označen plavom porukom *Programming Succeeded*. U suprotnom pojavljuje se crvena poruka *Programming Failed* i tada je potrebno ponoviti ceo postupak ponovo.



Slika 21: Početak programiranja FPGA čipa



Slika 22: Otvaranje bit fajla



Slika 23: Prozor nakon inicijalizacije Boundary-Scan Moda

Prilog 1: Veza pinova na ekspanzionim karticama sa sa pinovima Spartan2E FPGA kola u slučaju kada se povezuju na B1-B2 ekspanzioni konektor D2-SB kartice i A-B ekspanzioni konektor D2E kartice

Pin na ekspanzionoj kartici	FPGA pin na D2E kartici	FPGA pin na D2-SB kartici	Opis funkcije
BTN1	p40	p47	Tasteri
BTN2	p41	p46	
BTN3	p42	p86	
BTN4	p43	p84	
BTN5	p64	p83	
SW1	p16	p71	Prekidači
SW2	p18	p69	
SW3	p21	p64	
SW4	p23	p62	
SW5	p27	p60	
SW6	p30	p58	
SW7	p33	p56	
SW8	p35	p49	
LED1	p44	p111	Ledovke
LED2	p46	p109	
LED3	p48	p102	
LED4	p55	p100	
LED5	p57	p98	
LED6	p59	p96	
LED7	p61	p94	
LED8	p63	p89	
LEDG	p68	p88	Dozvola ledovki
SDP	p36	p48	Decimalna tačka
S0	p17	p70	Segmenti
S1	p20	p68	
S2	p22	p63	
S3	p24	p61	
S4	p29	p59	
S5	p31	p57	
S6	p34	p55	
AN1	p45	p82	Selekcija cifre na 7. segmentnom displeju
AN2	p47	p81	
AN3	p49	p75	
AN4	p56	p74	

Prilog 2: Veza pinova Spartan2E kola sa perifernim uređajima na razvojnoj kartici

Periferija na razvojnoj kartici	D2E kartica	D2-SB kartica	Opis funkcije
CLK	p80	p182	Glavni takt
LED1	p69	p154	Kontrolna LED
BTN1	p77	p187	Taster