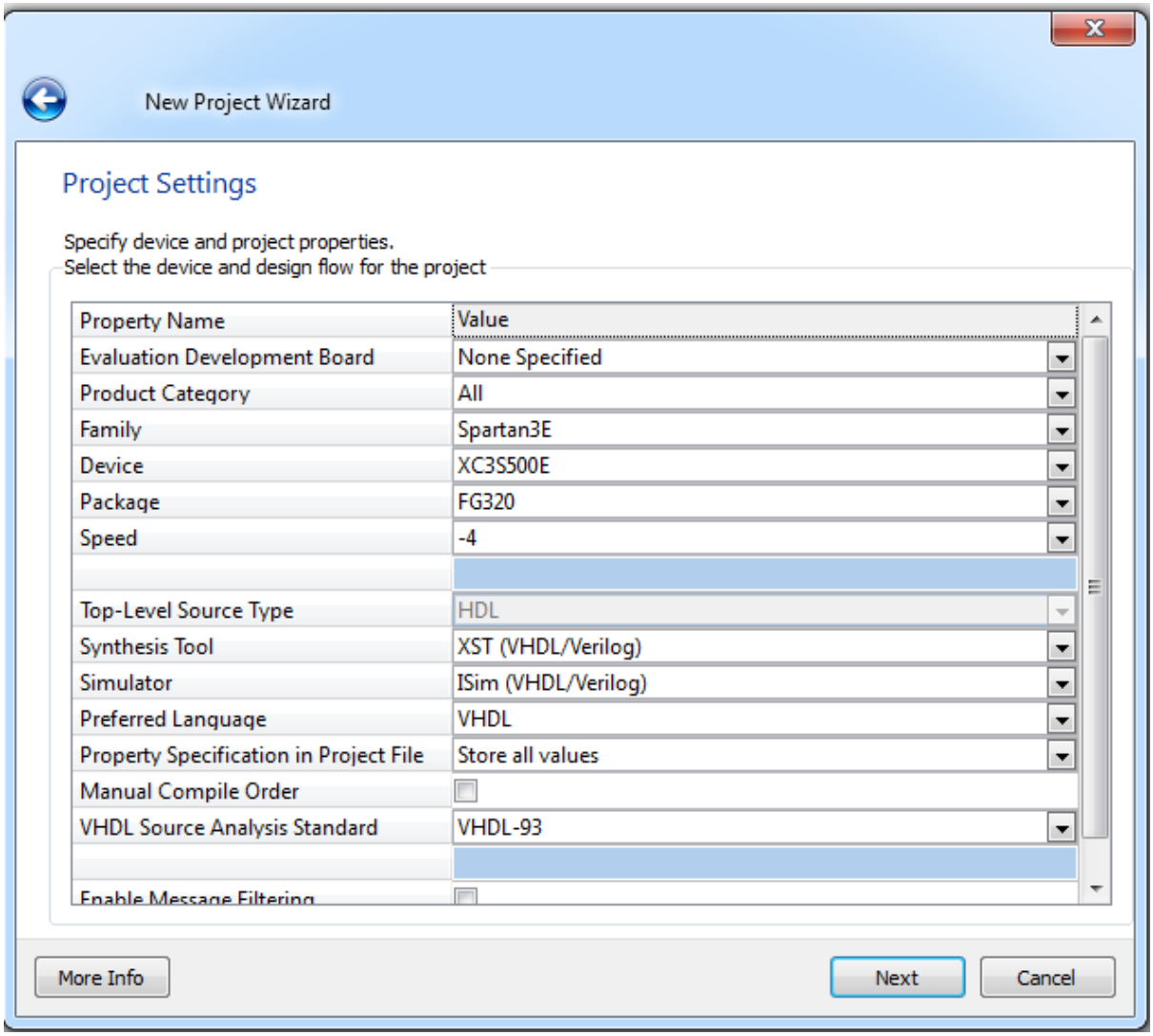
SIMULACIJA FUNKCIONALNOSTI DIZAJNA

Startovati **ISE Project Navigator** i kreirati novi projekat: **File->New Project**. Nakon unosa naziva projekta even_detector, potrebno je izvršiti određena podešavanja u okviru **Project Settings**, slika 1.



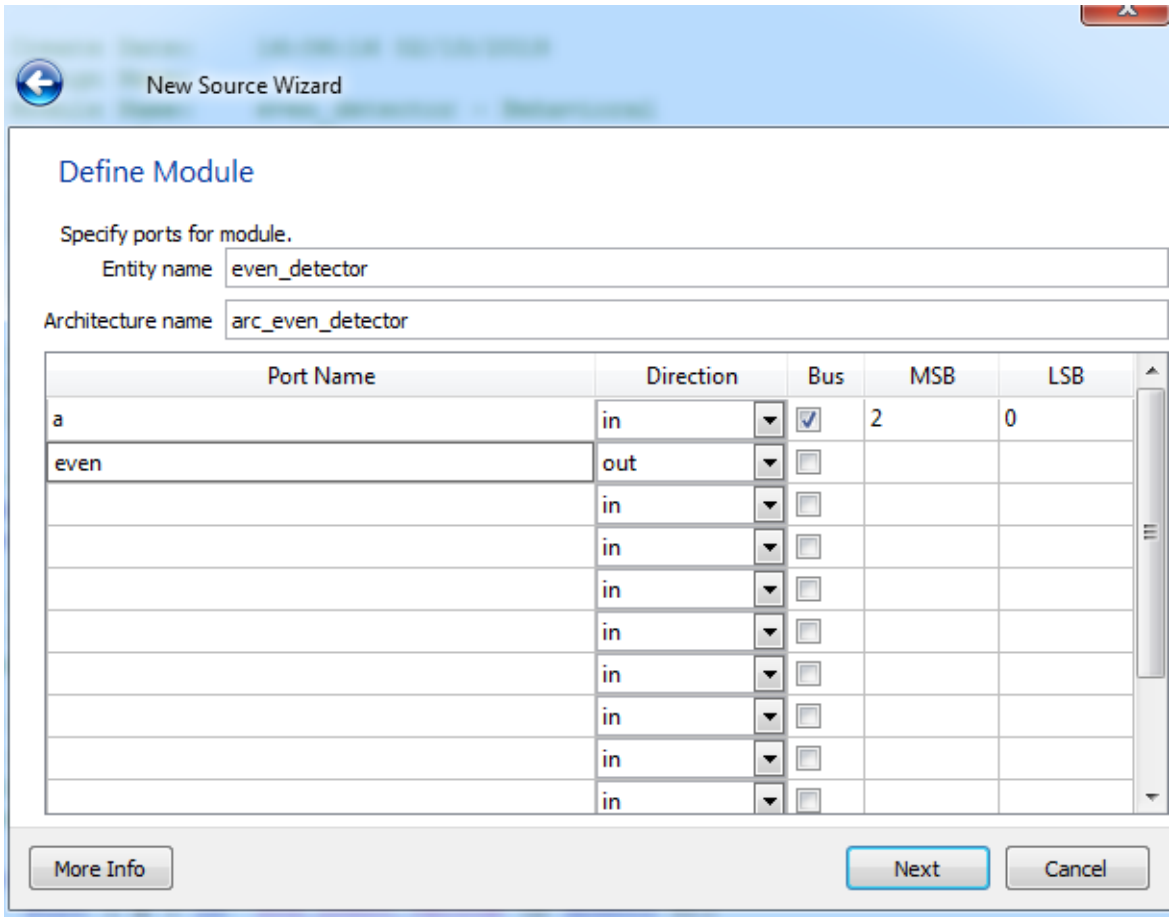
Slika 1. New Project Wizard

Izabrati <**Next>,** a potom <**Finish>**.

U okviry **Hierarchy**, izabrati even_detector, desni klik miša, a potpom **New Source**. U okviru **Select Source Type**, izabrati **VHDL Module**. Dodijeliti naziv even_detector, <**Next>**. Uz pomoć prikazanog **New Source Wizard**-a definisati modul kako je prikazano na slici 2.

Izabrati <**Next>**, a potom <**Finish>**.

Unijeti kod dat u listingu 1.

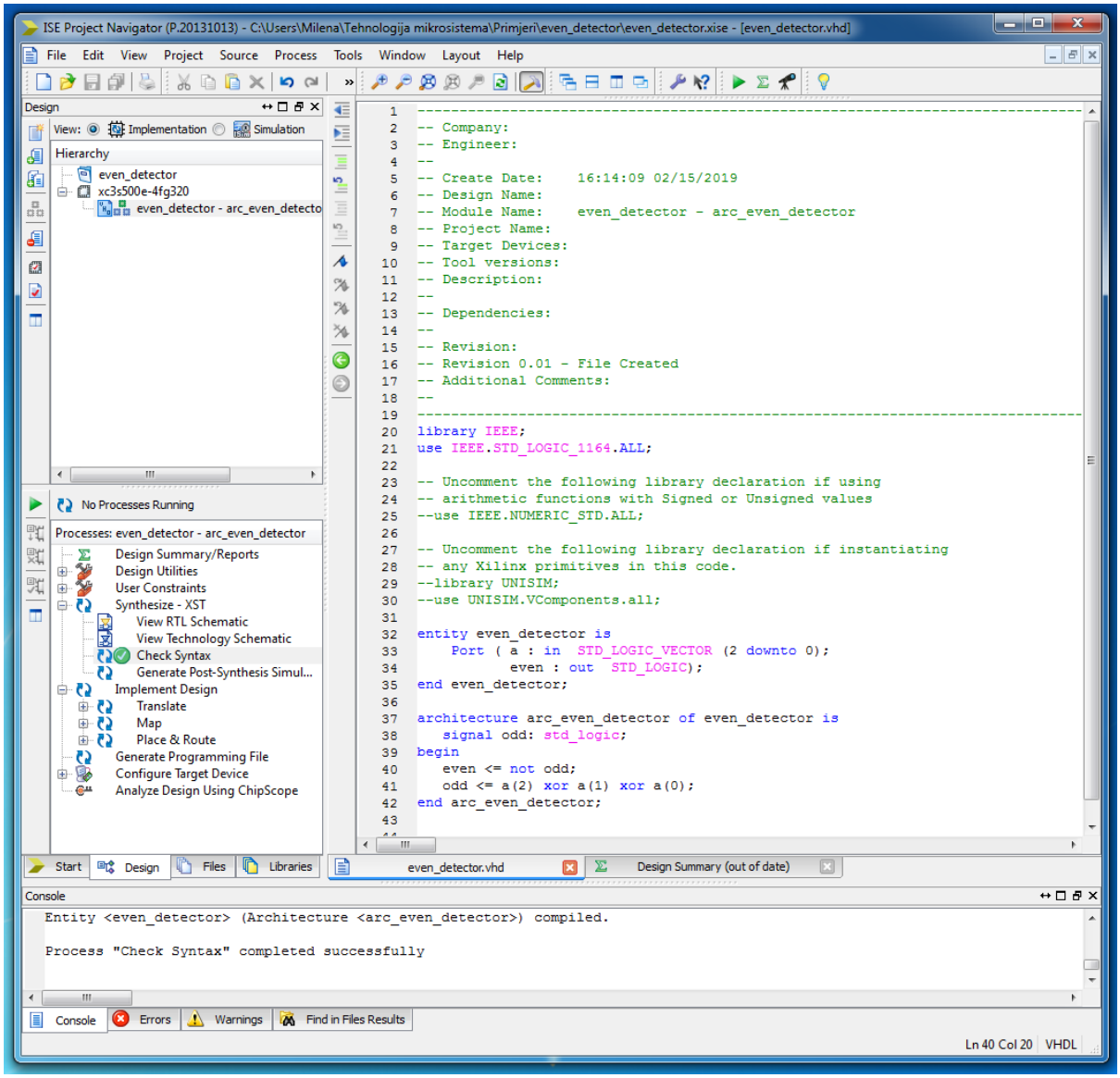


Slika 2. New Source Wizard

Listing 1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity even_detector is
    Port ( a : in  STD_LOGIC_VECTOR (2 downto 0);
           even : out  STD_LOGIC);
end even_detector;

architecture arc_even_detector of even_detector is
      signal odd: std_logic;
begin
      even <= not odd;
      odd <= a(2) xor a(1) xor a(0);
end arc_even_detector;
```

Radno okruženje je prikazano na slici 3. Izabrati opciju **Check Syntax**, kako bi izvršili provjeru sintakse koda.



Slika 3. Radno okruženje. Provejra sintakse koda.

U okviru **Hierarchy**, izabrati even_detector.vhd, desni klik miša, a potpom **New Source**. Izabrati **VHDL Test Bench**, dodijeliti naziv test_even_detector, <**Next**>, kao **Associate Source** izabrati even_detector, <**Next**>, <**Finish**>.

Unijeti kod dat u listingu 2. Nakon što je unijet odgovarajući kod, potrebno je u **Design** prozoru odabrati opciju **Simulaton** i pokrenuti simulaciju dvoklikom na **Simulate Behavioral Model**, slika 4.

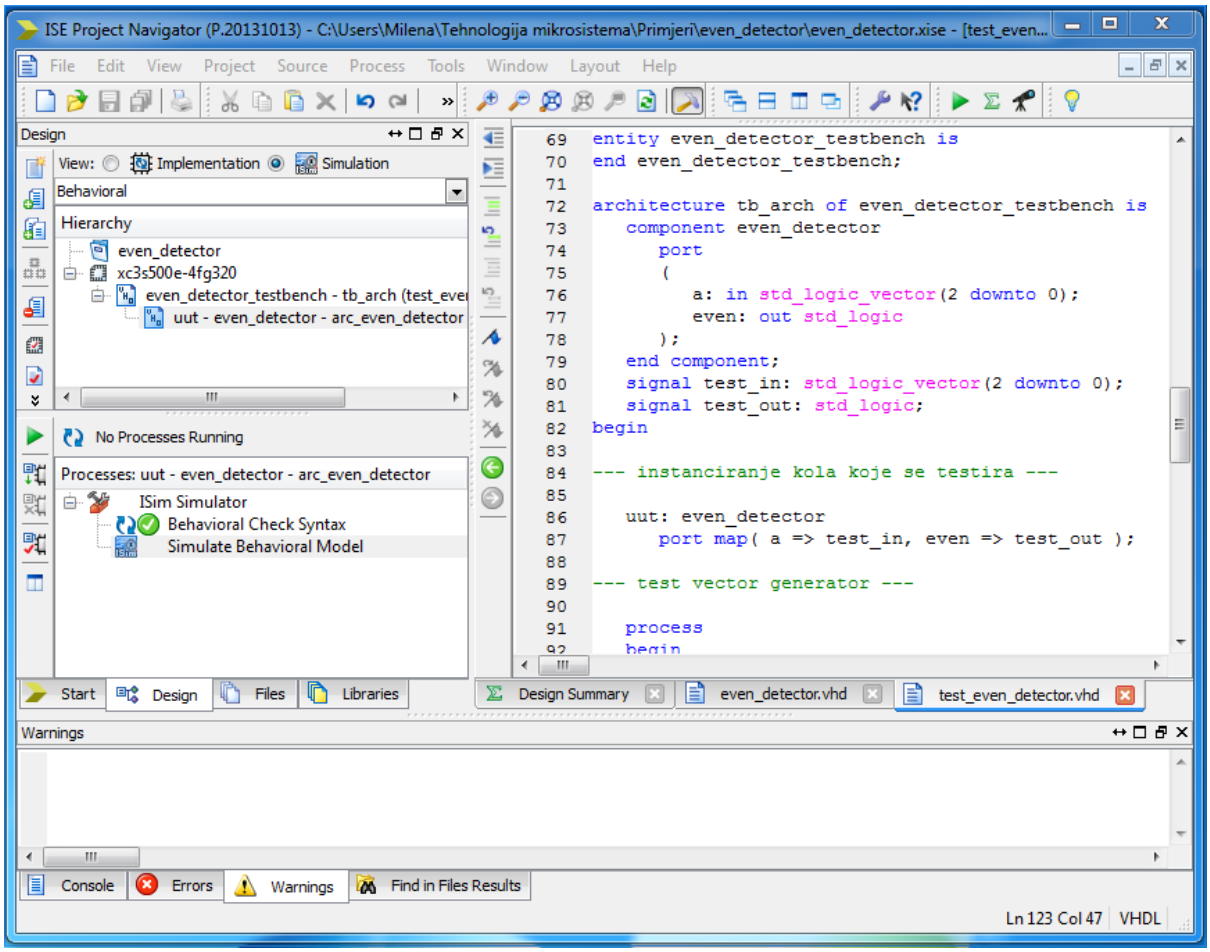
Nakon nekog vremena, prikazaće se prozor sa rezultatima simulacije, slika 5. U desnom dijelu, nalazi se dijagram sa talasnim oblicima signala. Kako je prikazano stanje signala na samom kraju simulacije, potrebno je vratiti se na njen početak pomjeranjem klizača koji se nalazi ispod dijagrama. Međutim, moguće je da razmjera na vremenskoj osi nije pogodna za pregled ovih signala pa je najkraći metod da se pritiskom na desni taster miša (dok se kursor nalazi na prostoru dijagrama) izabere opcija **To Full View**. Lijevo od dijagrama nalaze se dvije kolone: **Name** i **Value**. U koloni **Name** nalaze se imena signala čiji su dijagrami prikazani. Ovi signali se mogu rasporediti u proizvoljan redoslijed prostim prevlačenjem sa jedne pozicije na drugu pritiskom tastera miša. Moguće je i obrisati signale koji nisu od interesa.

U kolni **Value** se nalaze vrijednosti signala na poziciji na kojoj se trenutno nalazi kursor (žuta linija). Kursor se može pomjerati uz pritisak tastera miša.

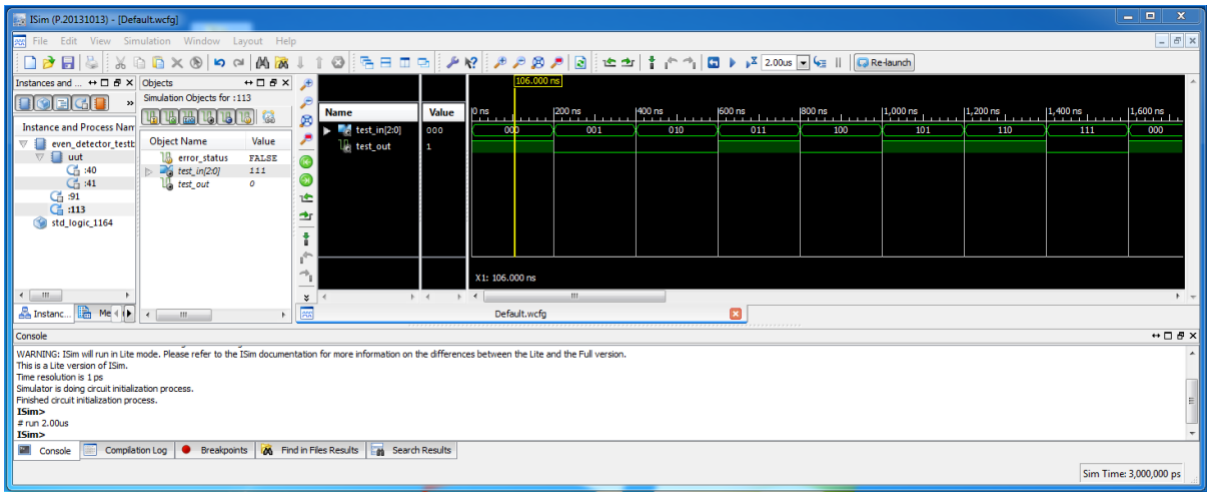
Listing 2

```
entity even_detector_testbench is
end even_detector_testbench;

architecture tb_arch of even_detector_testbench is
	component even_detector
		port
		(
			a: in std_logic_vector(2 downto 0);
			even: out std_logic
		);
	end component;
	signal test_in: std_logic_vector(2 downto 0);
	signal test_out: std_logic;
begin

--- instanciranje kola koje se testira ---

	uut: even_detector
		port map( a => test_in, even => test_out );

--- test vector generator ---

	process
	begin
		test_in <= "000";
		wait for 200 ns;
		test_in <= "001";
		wait for 200 ns;
		test_in <= "010";
		wait for 200 ns;
		test_in <= "011";
		wait for 200 ns;
		test_in <= "100";
		wait for 200 ns;
		test_in <= "101";
		wait for 200 ns;
		test_in <= "110";
		wait for 200 ns;
		test_in <= "111";
		wait for 200 ns;
	end process;

--- verifier ---

process
	variable error_status: boolean;
begin
	wait on test_in;
	wait for 100 ns;
	if ((test_in = "000" and test_out = '1') or
		(test_in = "001" and test_out = '0') or
		(test_in = "010" and test_out = '0') or
		(test_in = "011" and test_out = '1') or
		(test_in = "100" and test_out = '0') or
		(test_in = "101" and test_out = '1') or
		(test_in = "110" and test_out = '1') or
		(test_in = "111" and test_out = '0'))
	then
		error_status := false;
	else
		error_status := true;
	end if;

----- error reporting -----
	assert not error_status
```

```
            report "test failed."
            severity note;
        end process;
end tb_arch;
```



Slika 4. Radno okruženje. Simulacija funkcionalnog modela.



Slika 5. ISim Simulator. Simulacioni dijagrami.

Provjera ispravnosti rada modula se može izvršiti vizuelnim pregledom dijagrama, tako što se za svaku kombinaciju ulaznih signala provjeri vrijednost na izlazima.

Nakon završetka simulacije treba zatvoriti **ISim**. Pokretanje dvije instance simulatora nije dozvoljeno i često može da izazove probleme u radu.

Ukoliko se promijeni neki dio koda u **ISE** razvojnom okruženju, najlakše je restartovati simulaciju tako što će se zatvoriti **ISim** simulator, pa ponovo pokrenuti proces **Simulate Behavioral Model** unutar **ISE** razvojnog okruženja.

Umjesto vizuelnog pregleda dijagrama nakon svake simulacije, moguće je napisati testni kod koji će automatski provjeriti ispravnost izlaznih signala. Za svaki izlazni signal može se napisati task koji prihvata ulazni parametar (koji sadrži očekivanu vrijednost izlaza) i provjerava da li se njegova vrijednost poklapa sa trenutnim stanjem na izlazu. Ako tokom simulacije dođe do grešaka, na konzoli će se ispisati vremenski trenutak kada se greška dogodila, trenutna vrijednost izlaznog signala i vrijednost koja se očekivala. Ako nije bilo grešaka, na konzoli se neće ništa ispisivati.

VREMENSKA SIMULACIJA DIZAJNA

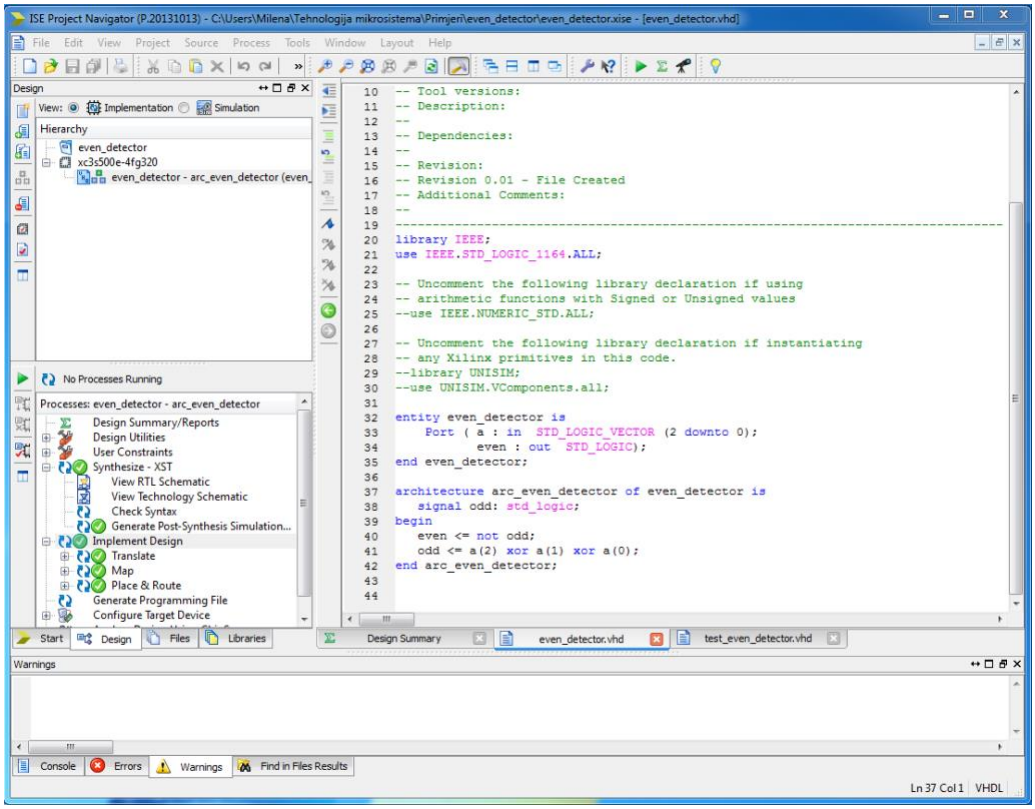
Osim funkcionalne ispravnosti dizajna, neophodno je voditi računa i o ostalim aspektima koji utiču na ispravno funkcionisanje uređaja koji dizajniramo. Jedan od važnih faktora je i vrijeme koje je potrebno za propagaciju pojedinih signala kroz FPGA kolo. Stoga je potrebno izvršiti i tzv. vremensku simulaciju dizajna kako bi se procijenilo kojom se brzinom prostiru signali i na taj način izbjeglo nepredviđeno ponašanje nakon upisa dizajna u FPGA čip.

Da bi se izvršila vremenska simulacija i analiza, **ISE** mora izvršiti procese synthesize, translate, map i place & route. U tu svrhu je potrebno izabrati opciju **Implementation**, slika 6. Navedeni procesi se nalaze unutar grupa **Synthesize – XST** i **Implement Design**.

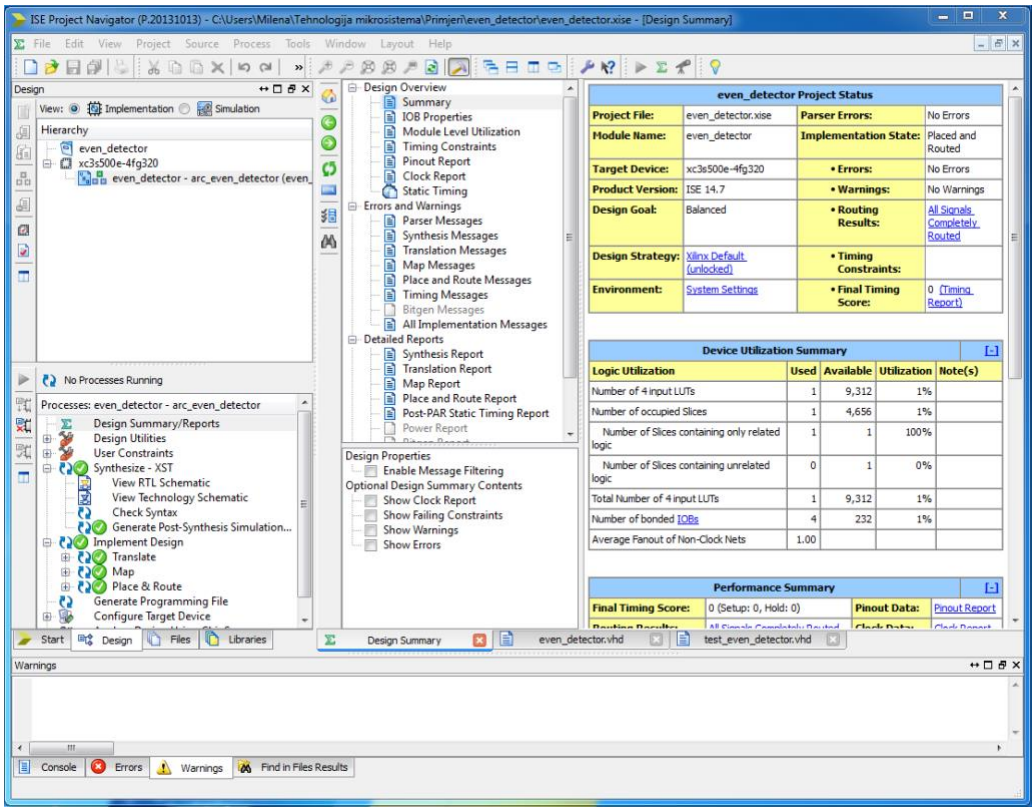
Dvostrukim klikom na **Design Summary/Reports** dobija se detaljan izvještaj o rezultatu procesa dizajna, slika 7.

Unutar sekcije **Detailed Reports** može se izabrati **Post PAR Static Timing Report**, kako bi se vidjelo kašnjenje koje nastaje zbog propagacije signala, slika 8.

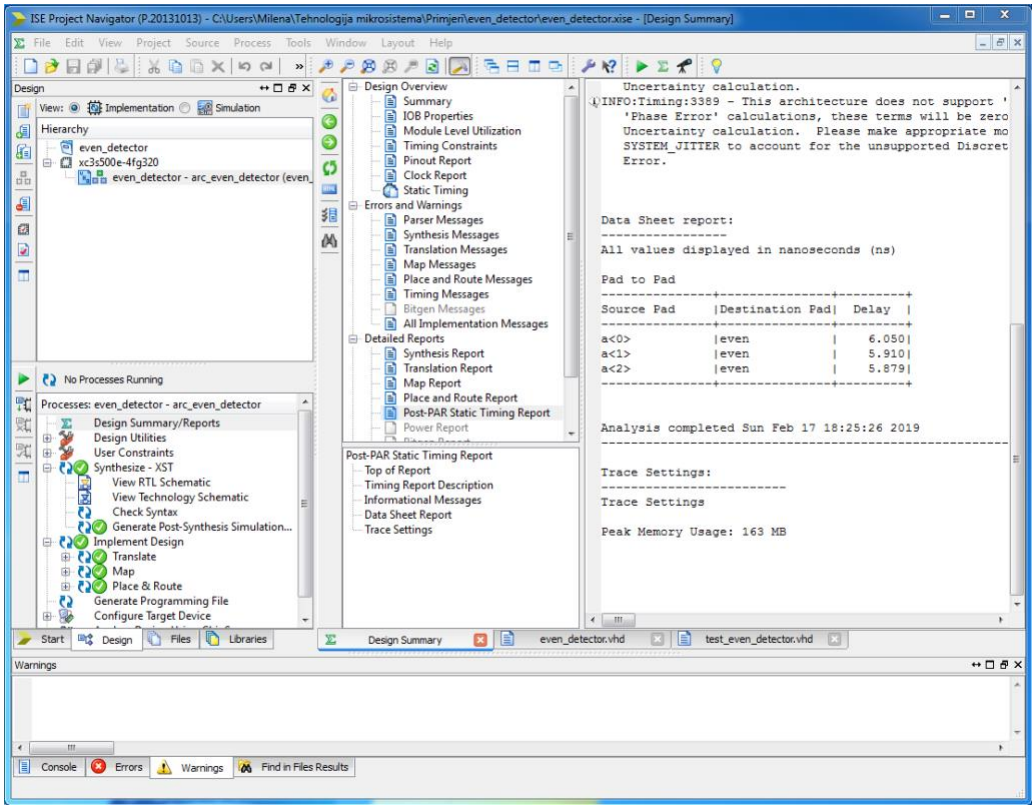
Unutar sekcije **Synthesis Report** se može naći informacija o maksimalnom kašnjenju kombinacionih kola, koje u datom primjeru iznosi 6,209 ns, slika 9. Ukoliko bi se ulazni signali mijenjali brže od ove vrijednosti, moglo bi doći do neočekivanog ponašanja našeg dizajna.



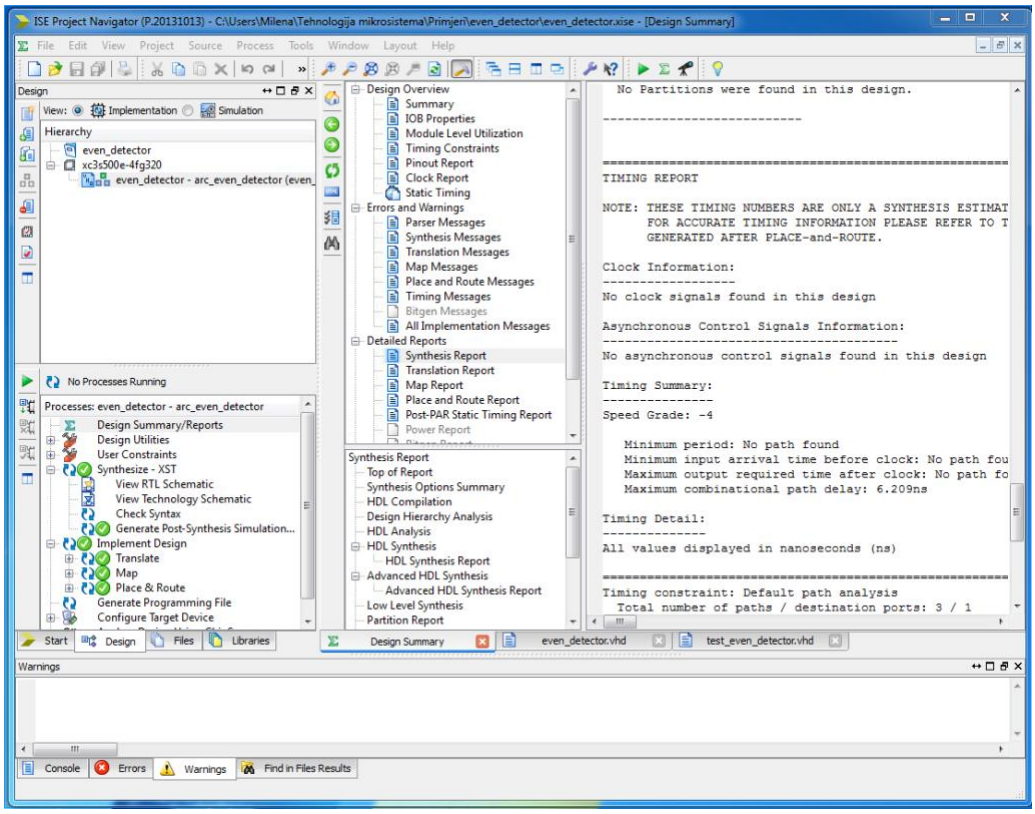
Slika 6. Radno okruženje. Implementacija.



Slika 7. Radno okruženje. Pregled karakteristika dizajna.



Slika 8. Radno okruženje. Propagaciono kašnjenje.



Slika 9. Radno okruženje. Maksimalno kašnjenje kombinacionih kola.