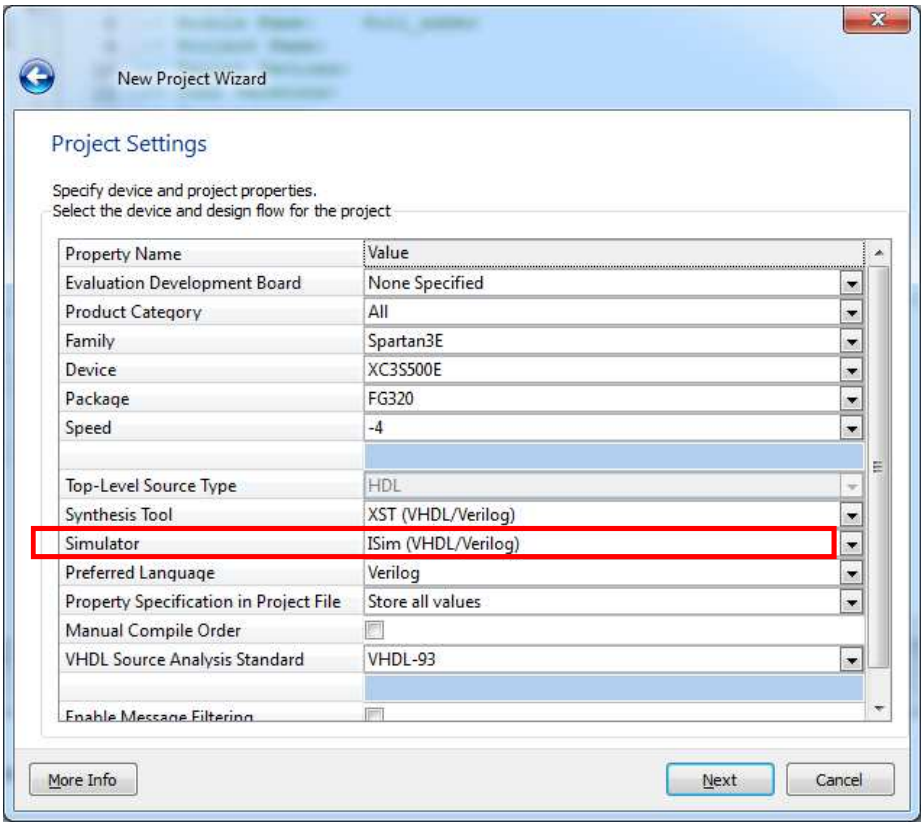
ISim simulator – uputstvo za upotrebu

ISim simulator je alat koji je integrisan u Xilinx ISE razvojno okruženje kako bi se omogućilo testiranje dizajna. Pomoću ISim simulatora se obavlja ispitivanja funkcionalnosti dizajna (tzv. *behavioral simulation*). Osim ovoga, potrebno je uzeti u obzir i vremenske karakteristike logičkih sklopova unutar FPGA kola, kako bi se utvrdilo koliko dugo se signali propagiraju i kako bi bili sigurni da će se dizajn ponašati kako je očekivano i kada se unese u FPGA čip.

Simulacija funkcionalnosti dizajna

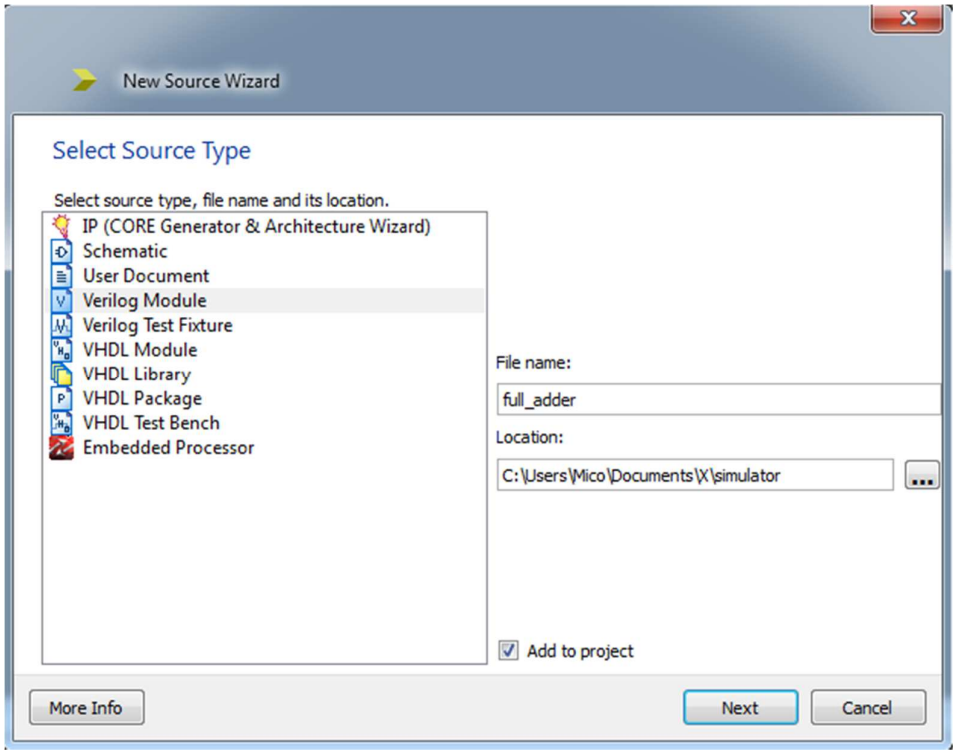
Startovati ISE Project Navigator i kreirati novi projekat: *File->New Project*. Nakon unosa imena projekta, pojavljuje se prozor za konfiguraciju projekta (*Project Settings*):



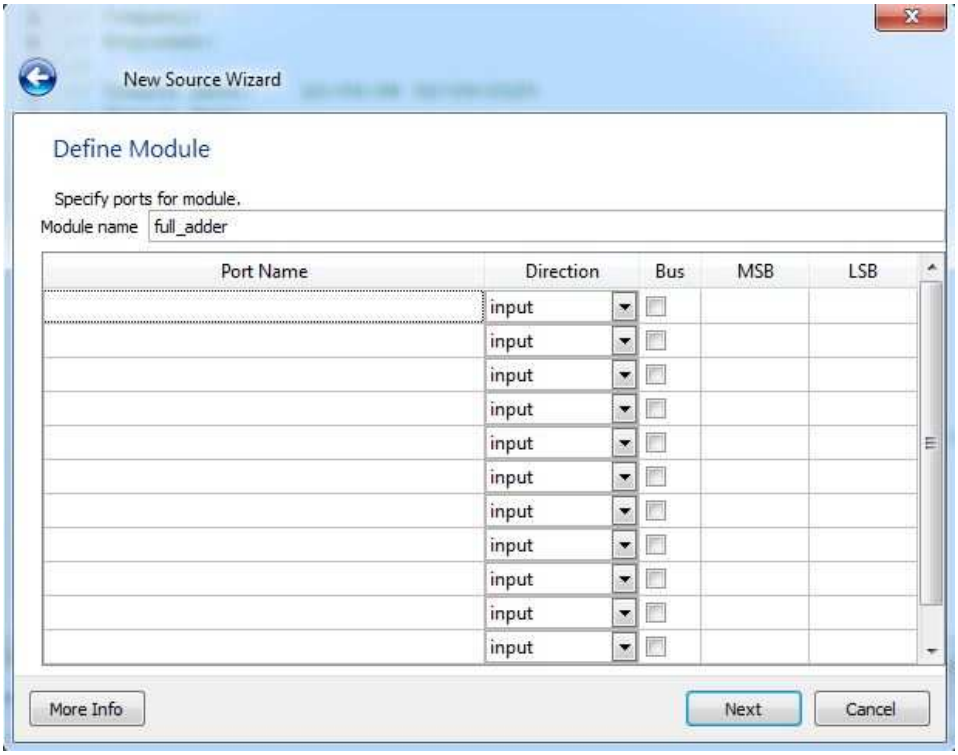
Slika 1

Potrebno je izabrati ISim kao podrazumijevani simulator (vidi sliku 1).

Kliknuti *Next->Finish* i unutar ovog projekta kreirati novi Verilog modul (*Project->New Source->Verilog Module*) sa imenom *full_adder* (slika 2). Nakon klika na *Next* pojaviće se prozor za definisanje modula (slika 3). U ovom slučaju nećemo koristiti mogućnost da u ovom prozoru definišemo nazive portova, njihov smjer i širinu, pa ćemo samo kliknuti na *Next* i *Finish*.



Slika 2



Slika 3

Verilog kod za ovaj modul izgleda ovako:

```
module full_adder(s, cout, a, b, cin);
    output s, cout;
    input a, b, cin;

    wire t1, t2, t3;
```

```
    xor (t1, a, b);
    xor (s, t1, cin);
    and (t2, t1, cin);
    and (t3, a, b);
    or  (cout, t2, t3);
endmodule
```

Da bi bili sigurni da je kod ispravno unijet možemo pokrenuti *Check Syntax* proces (unutar *Synthesize* opcije) i nakon toga snimiti projekat.

Kliknuti desnim tasterom na *full_adder.v* u okviru *Hierarchy* prozora i izabrati *New Source*. Izabrati *Verilog Test Fixture* i dati ime novom fajlu "*test_full_adder*". Kliknuti *Next*, i pojaviće se zahtjev da se izvrši pridruživanje ovog fajla sa modulom. Izabrati *full_adder*, kliknuti *Next*, a zatim *Finish*. Fajl će biti dodat projektu.

ISE kreira kostur koda za testiranje. Modul *full_adder* je instanciran i simulacija počinje nakon "*initial begin*" linije. Promjenljive su inicijalizovane i kod koji treba dodati ide nakon "*Add stimulus here*" linije:

```
module test_full_adder;
      // Inputs
      reg a;
      reg b;
      reg cin;

      // Outputs
      wire s;
      wire cout;

      // Instantiate the Unit Under Test (UUT)
      full_adder uut (
            .s(s),
            .cout(cout),
            .a(a),
            .b(b),
            .cin(cin)
      );

      initial begin
            // Initialize Inputs
            a = 0;
            b = 0;
            cin = 0;

            // Wait 100 ns for global reset to finish
            #100;

            // Add stimulus here
      end
endmodule
```

Modul se testira tako što se različite vrijednosti dodjeljuju ulazima i posmatraju se vrijednosti na izlazima. Između dodjeljivanja vrijednosti umeću se izvjesna kašnjenja korišćenjem ***#n***, gdje je ***n*** iznos kašnjenja u nanosekundama. Na primjer, sledeći kod testira ponašanje kola kad je po jedan ulaz na visokom logičkom nivou, pri čemu se promjena vrijednosti na ulazima vrši svakih 25ns:

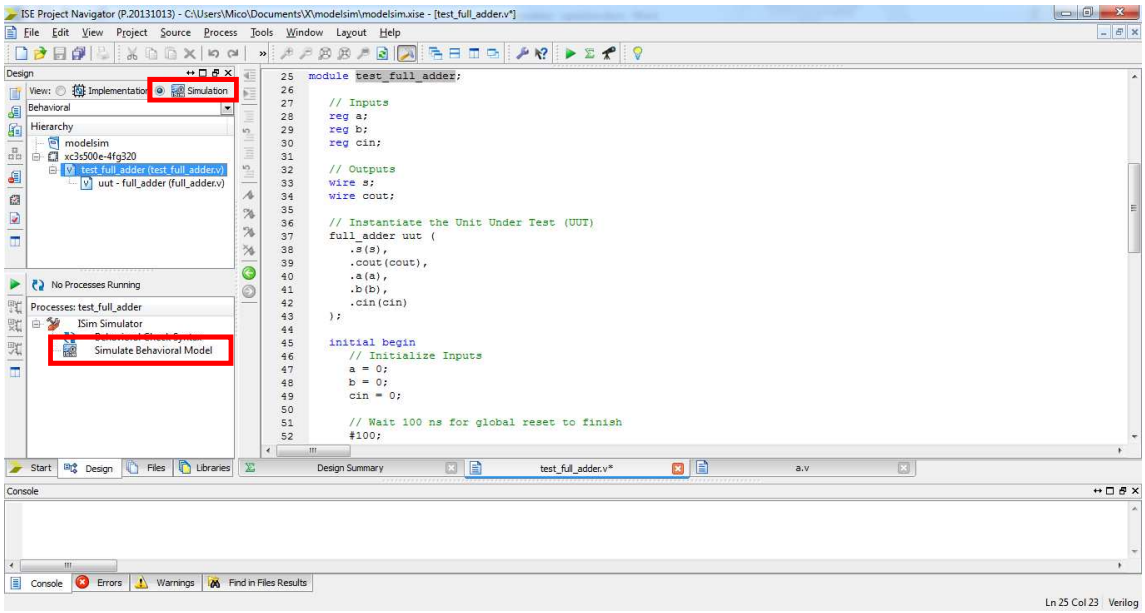
```
// Add stimulus here
//125 ns
#25;    a = 1'b1;
//150 ns
#25;    a = 1'b0;   b = 1'b1;
//175 ns
#25;    b = 1'b0;   cin = 1'b1;
```

Napomena: za vrijednost **1'b0**, jedinica predstavlja broj bitova, **b** znači da je u pitanju binarni broj, a **0** predstavlja jednobitnu binarnu vrijednost. Vrijednosti se mogu predstavljati i u decimalnom i heksadecimalnom brojnom sistemu. Na primjer, **8'd27** predstavlja **00011011** a **4'hA** prestavlja **1010**.

Kvalitetan test mora testirati sve ili bar veći broj mogućih ulaznih kombinacija svakog korišćenog modula, sa posebnim osvrtom na granične/osjetljive slučajeve. Pošto potpuni binarni sabirač ima tri jednobitna ulaza, postoji osam mogućih kombinacija na ulazima i test bi trebalo da sve njih simulira. Odgovarajući kod izgleda ovako:

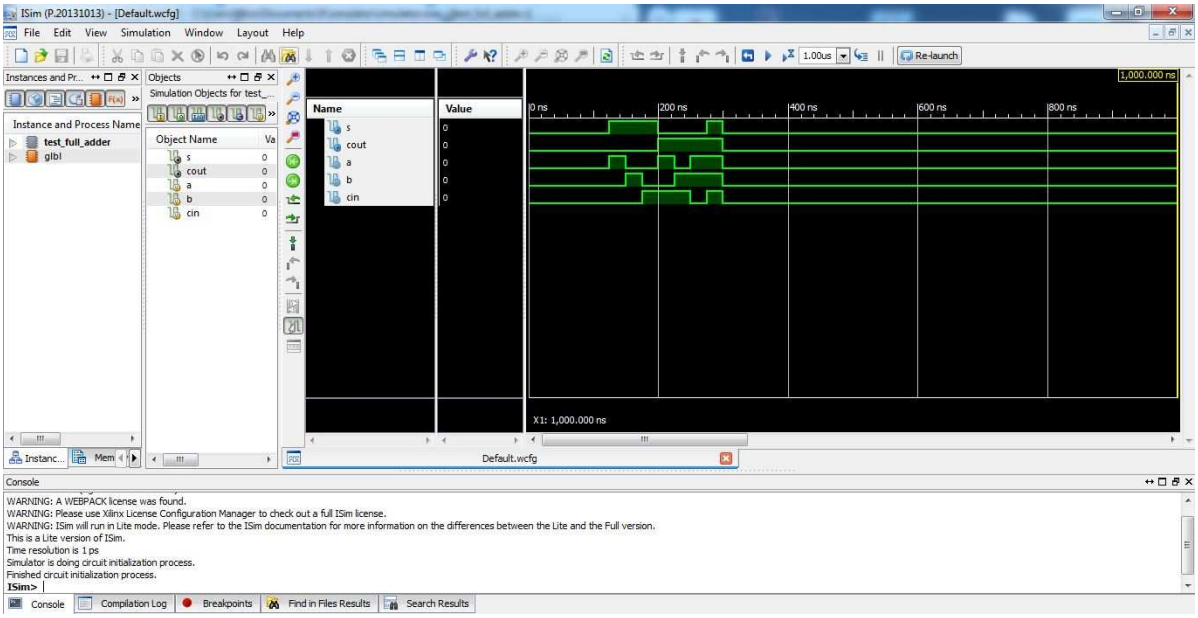
```
//125 ns
#25;    a = 1'b1;
//150 ns
#25;    a = 1'b0;   b = 1'b1;
//175 ns
#25;    b = 1'b0;   cin = 1'b1;
//200 ns
#25;    a = 1'b1;
//225 ns
#25;    a = 1'b0;   b = 1'b1;
//250 ns
#25;    a = 1'b1;   cin = 1'b0;
//275 ns
#25;    cin = 1'b1;
//300 ns
#25;    a = 1'b0;   b = 1'b0;   cin = 1'b0;
```

Nakon što je kod unijet u odgovarajući fajl, potrebno je u *Design* prozoru odabrati opiju *Simulation*, u *Hierarchy* prozoru selektovati *test_full_adder*, pa pokrenuti simulaciju dvoklikom na *Simulate Behavioral Model* (slika 4):



Slika 4

Poslije nekog vremena pojaviće se novi prozor sa rezultatima simulacije (ISim). U desnom dijelu prozora nalazi se dijagram sa talasnim oblicima svih ulaznih i izlaznih signala. Pošto 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"*. Isti efekat se postiže i pomoću tastera F6:



Slika 5

Za ovaj dizajn, skala koja nas interesuje je između 0 i 300ns.

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.

Provjera ispravnosti rada modula se može izvršiti vizuelnim pregledom dijagrama, tako što se za svaku kombinaciju ulaznih signala provjeri vrijednost na izlazima. U tom procesu može pomoći pomjeranje kursora kako bi se posmatrale numeričke vrijednosti signala u koloni *Values* umjesto da se posmatraju dijagrami. Kod modula sa vektorskim signalima (višebitnim signalima) može biti korisno da se signali prikazuju u nekom drugom brojnom sistemu (*Radix*): dekadnom, heksadecimalnom, …

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.

Napredna simulacija funkcionalnosti dizajna

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 se vrijednosti ne poklapaju ispisuje se poruka o grešci na simulacionoj konzoli. Kod za *task*-ove za ***s*** i ***cout*** izgleda ovako:

```
    task CHECK_s;
        input NEXT_s;
        #0 begin
            if (NEXT_s !== s) begin
                $display("Error at time=%dns s=%b, expected=%b", $time,
s, NEXT_s);
                end
            end
        endtask
    task CHECK_cout;
        input NEXT_cout;
        #0 begin
            if (NEXT_cout !== cout) begin
                $display("Error at time=%dns cout=%b, expected=%b",
$time, cout, NEXT_cout);
                end
            end
        endtask
```

Ovi *task*-ovi se pozivaju u simulacionom bloku, neko vrijeme nakon što se postave vrijednosti ulaznih signala, kako bi se dozvolila njihova propagacija kroz dizajn.

Kompletan kod *„test_full_adder.v“* sada izgleda ovako:

```
module test_full_adder;

      // Inputs
      reg a;
      reg b;
      reg cin;

      // Outputs
      wire s;
      wire cout;

      // Instantiate the Unit Under Test (UUT)
      full_adder uut (
            .s(s),
            .cout(cout),
            .a(a),
            .b(b),
            .cin(cin)
      );

      initial begin
            // Initialize Inputs
            a = 0;
            b = 0;
            cin = 0;

            // Wait 100 ns for global reset to finish
            #100;

            // Add stimulus here
        #25;    a = 1'b1;
        //150 ns
```

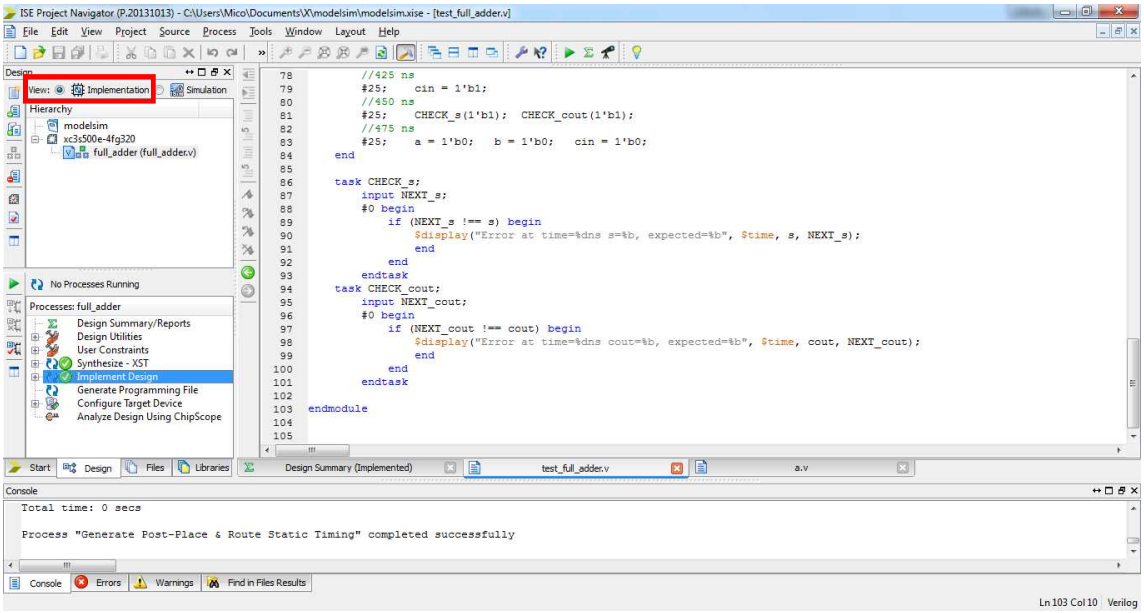
```
        #25;    CHECK_s(1'b1);  CHECK_cout(1'b0);
        //175 ns
        #25;    a = 1'b0;   b = 1'b1;
        //200 ns
        #25;    CHECK_s(1'b1);  CHECK_cout(1'b0);
        //225 ns
        #25;    b = 1'b0;   cin = 1'b1;
        //250 ns
        #25;    CHECK_s(1'b1);  CHECK_cout(1'b0);
        //275 ns
        #25;    a = 1'b1;
        //300 ns
        #25;    CHECK_s(1'b0);  CHECK_cout(1'b1);
        //325 ns
        #25;    a = 1'b0;   b = 1'b1;
        //350 ns
        #25;    CHECK_s(1'b0);  CHECK_cout(1'b1);
        //375 ns
        #25;    a = 1'b1;   cin = 1'b0;
        //400 ns
        #25;    CHECK_s(1'b0);  CHECK_cout(1'b1);
        //425 ns
        #25;    cin = 1'b1;
        //450 ns
        #25;    CHECK_s(1'b1);  CHECK_cout(1'b1);
        //475 ns
        #25;    a = 1'b0;   b = 1'b0;   cin = 1'b0;
    end

    task CHECK_s;
        input NEXT_s;
        #0 begin
            if (NEXT_s !== s) begin
                $display("Error at time=%dns s=%b, expected=%b", $time,
s, NEXT_s);
                end
            end
        endtask
    task CHECK_cout;
        input NEXT_cout;
        #0 begin
            if (NEXT_cout !== cout) begin
                $display("Error at time=%dns cout=%b, expected=%b",
$time, cout, NEXT_cout);
                end
            end
        endtask

endmodule
```

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.

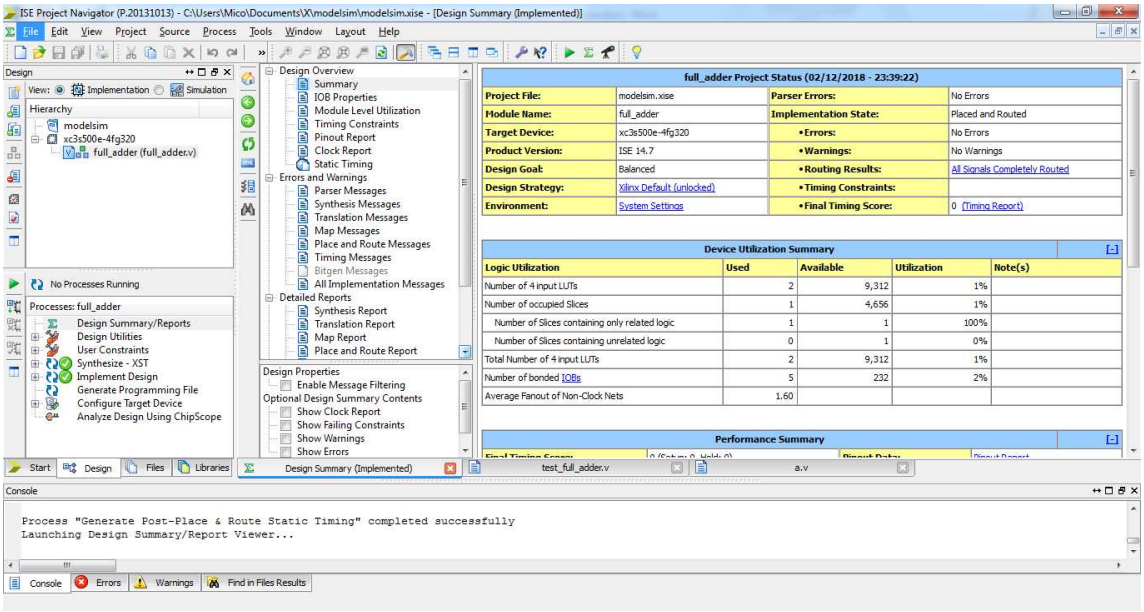
Da bi se izvršila vremenska simulacija i analiza, ISE mora izvršiti procese ***synthesize***, ***translate***, ***map*** i ***place & route***. U tu svrhu je potrebno izabrati opciju *Implementation* (slika 6), pa dvostruki klik na *Implement Design*:



Slika 6

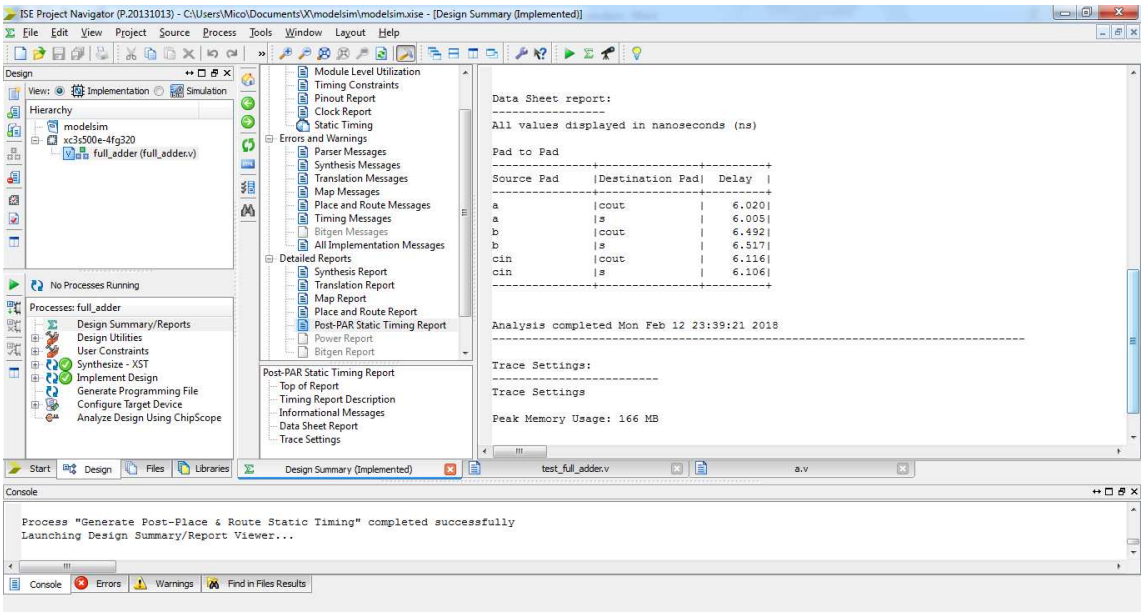
Navedeni procesi se nalaze unutar grupa ***Synthesize – XST*** i ***Implement Design***.

Dvostrukim klikom na ***Design Summary/Reports*** dobija se detaljan izvještaj o rezultatu procesa dizajna:



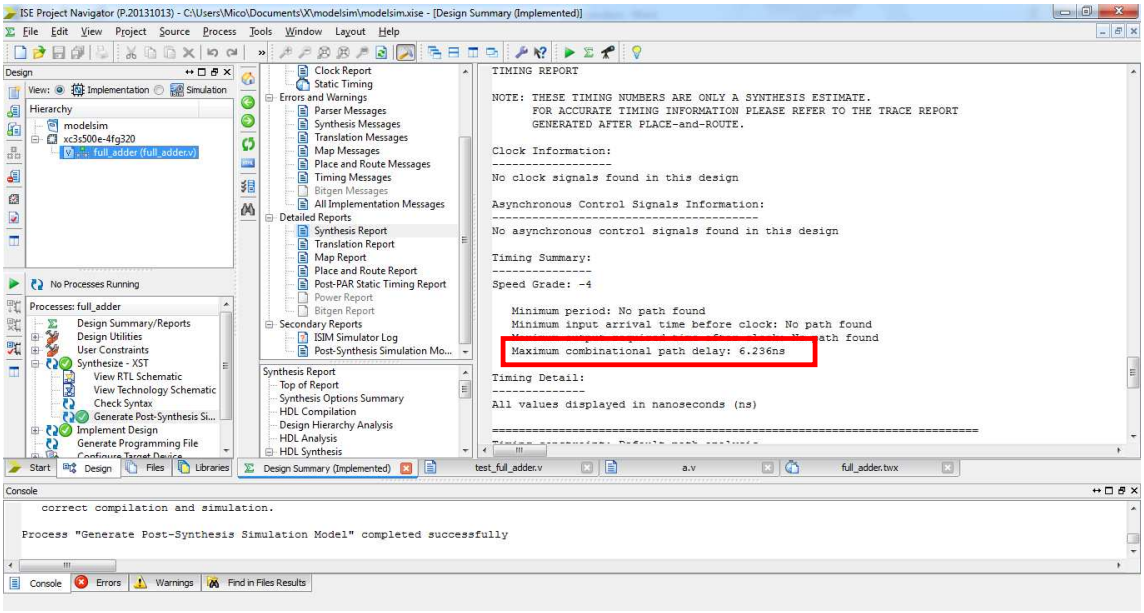
Slika 7

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



Slika 8

Unutar sekcije ***Synthesize Report*** se može naći informacija o maksimalnom kašnjenju kombinacionih kola, koje u našem primjeru iznosi 6,236ns (slika 9).



Slika 9

Ukoliko bi se ulazni signali mijenjali brže od ove vrijednosti, moglo bi doći do neočekivanog ponašanja našeg dizajna.

Vremenska simulacija dizajna kod sekvencijalnih kola

Za primjer sekvencijalnog kola uzećemo dvobitni brojač:

```
`timescale 1ns / 1ps

module counter(CLK, IN, OUT);
```

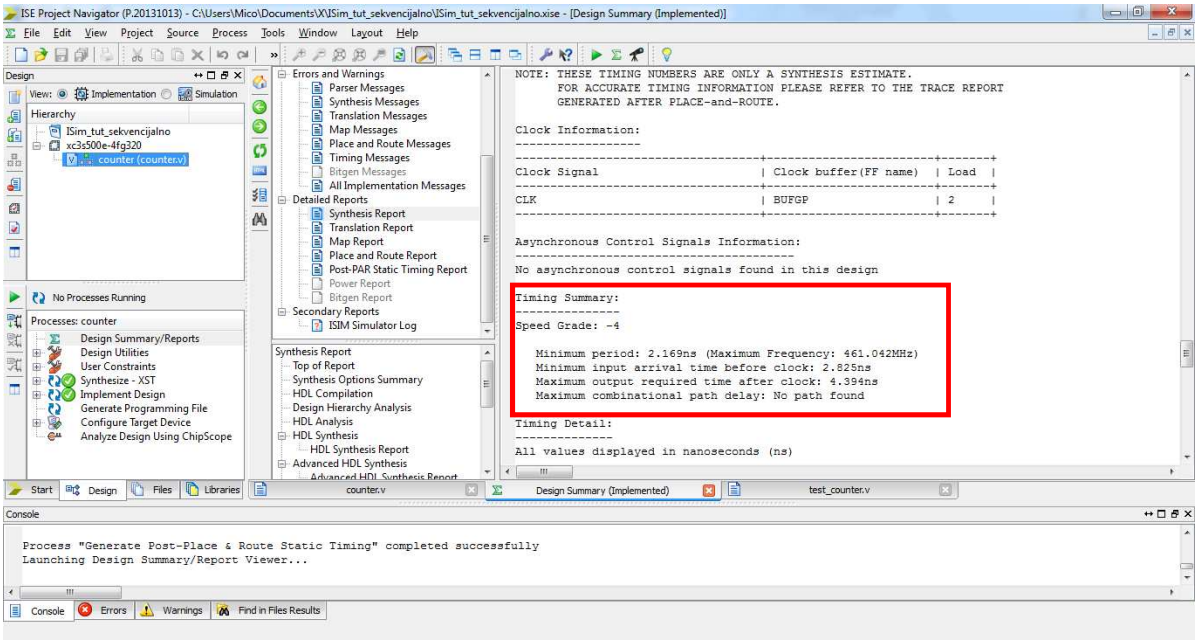
```
    input         CLK;
    input         IN;
    output [1:0] OUT;

    reg [1:0] OUT = 0;
    always @(posedge CLK) begin
        OUT = IN ? OUT-1 : OUT+1;
        end

endmodule
```

U novi projekat treba unijeti ovaj fajl i pridružiti mu *Verilog test fixture* u kome će se izvan *initial* bloka nalaziti sledeći kod:

```
always #5 CLK <= ~CLK;
```

Na ovaj način će se promjena taktnog signala vršiti svakih 5ns, što znači da će period biti 10ns, odnosno frekvencija takta 100MHz.

Nakon ponavljanja iste procedure kao u prethodnom slučaju, otvoriti ***Synthesis Report*** i pronaći ***Timing Summary*** sekciju (slika 10).



Slika 10

Tu se nalazi informacija o minimalnom periodu: 2,169ns (Maksimalna frekvencija 461,042MHz). Korišćenjem taktnog signala periode manje od 2,169ns dovešće do nepredvidljivog ponašanja brojača.