
# Laborator 2

## 1. Scopul laboratorului

Acest laborator reprezinta o introducere in mediul de dezvoltare Xilinx.

Xilinx ISE 10.1 este folosit pentru a implementa pe un FPGA circuitul descris in limbajul Verilog.

Acest laborator contine un ghid de creare a unui proiect Xilinx ce are ca scop realizarea functiei xor prin folosirea a doua switchuri si a unui led aflate pe placa Spartan 3E Starter Kit.

Ulterior, aplicatia din laboratorul anterior va fi testata pe placa prin aprinderea a doua leduri la frecventele corespunzatoare.

# 2.1 Crearea unui proiect nou

- Pentru pornirea aplicatiei se face dublu-click pe icoana Xilinx ISE 10.1 (Figura 2.1).


Figura 2.1

- File –> New Project (Figura 2.2)

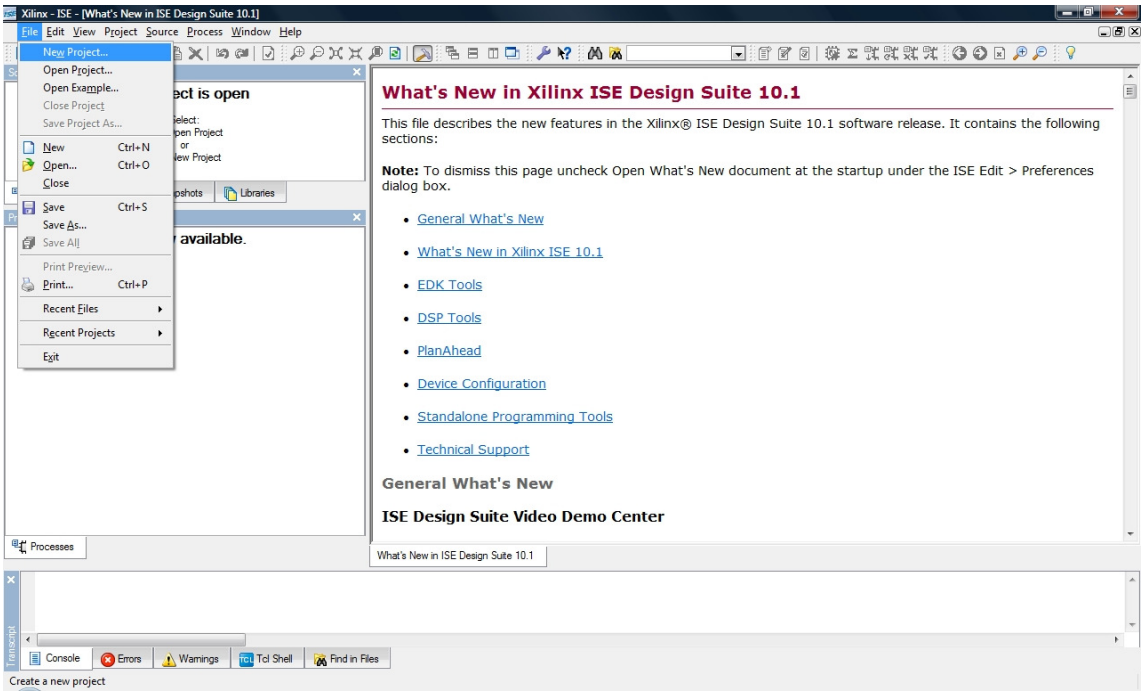


Figura 2.2

- In fereastra urmatoare (Figura 2.3) vor trebui completate: numele proiectului „laboratorAC1”, locatia si tipul sursei „HDL”.

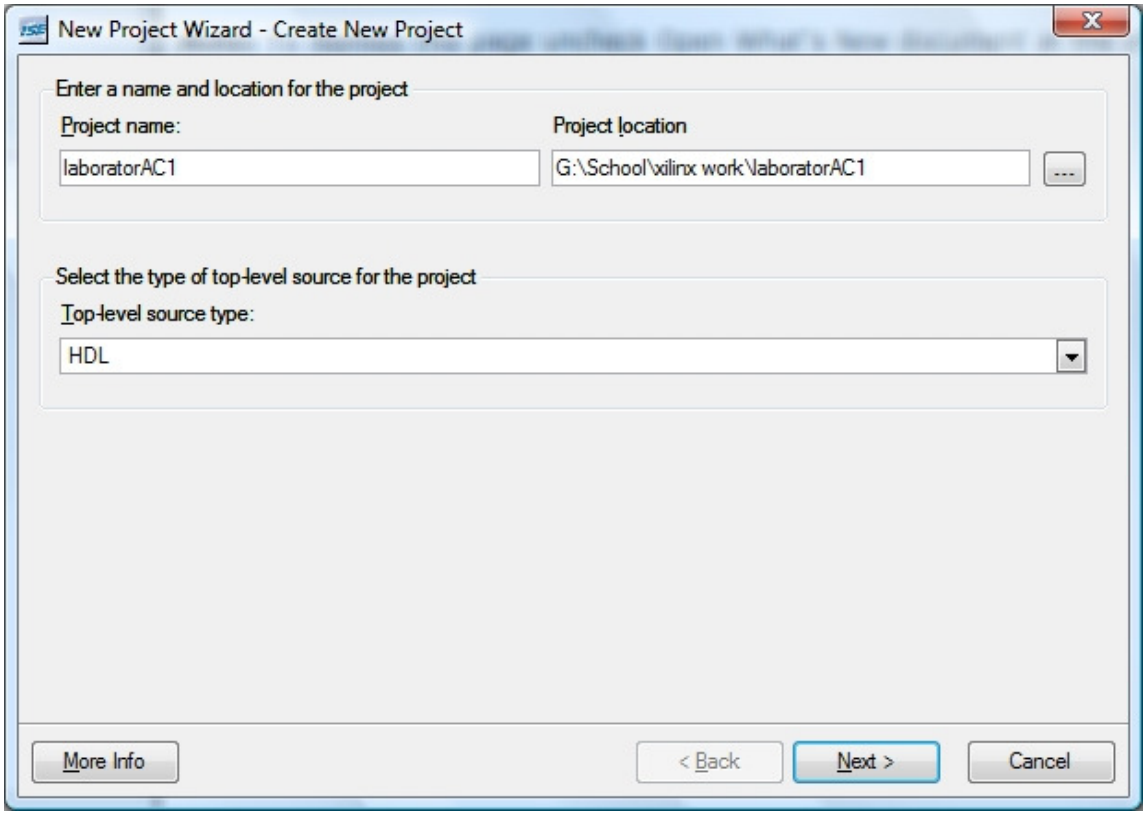


Figura 2.3

- La urmatorul pas se vor completa campurile din Figura 2.4 cu optiunile prezentate:

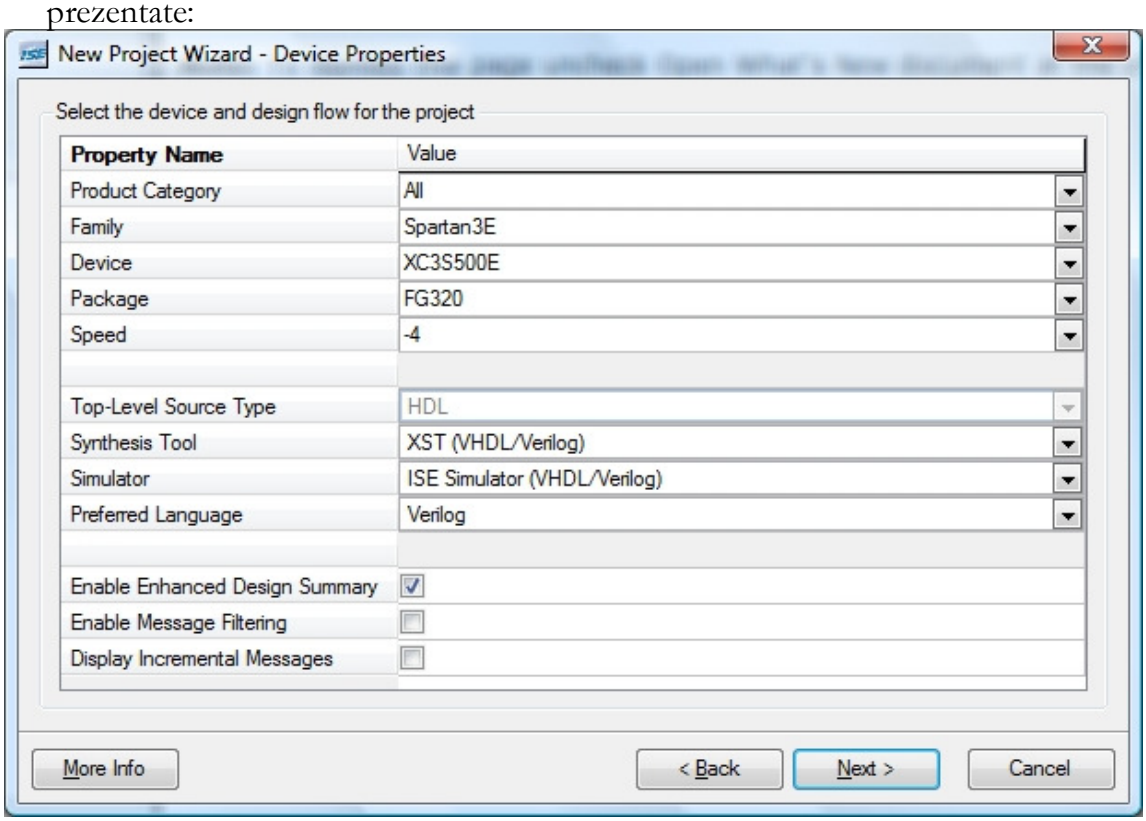


Figura 2.4

- Urmatoarele doua ferestre (Figura 2.5 si 2.6) dau posibilitatea adaugarii de fisiere sursa existente. In acest caz vor fi lasate necompletate.

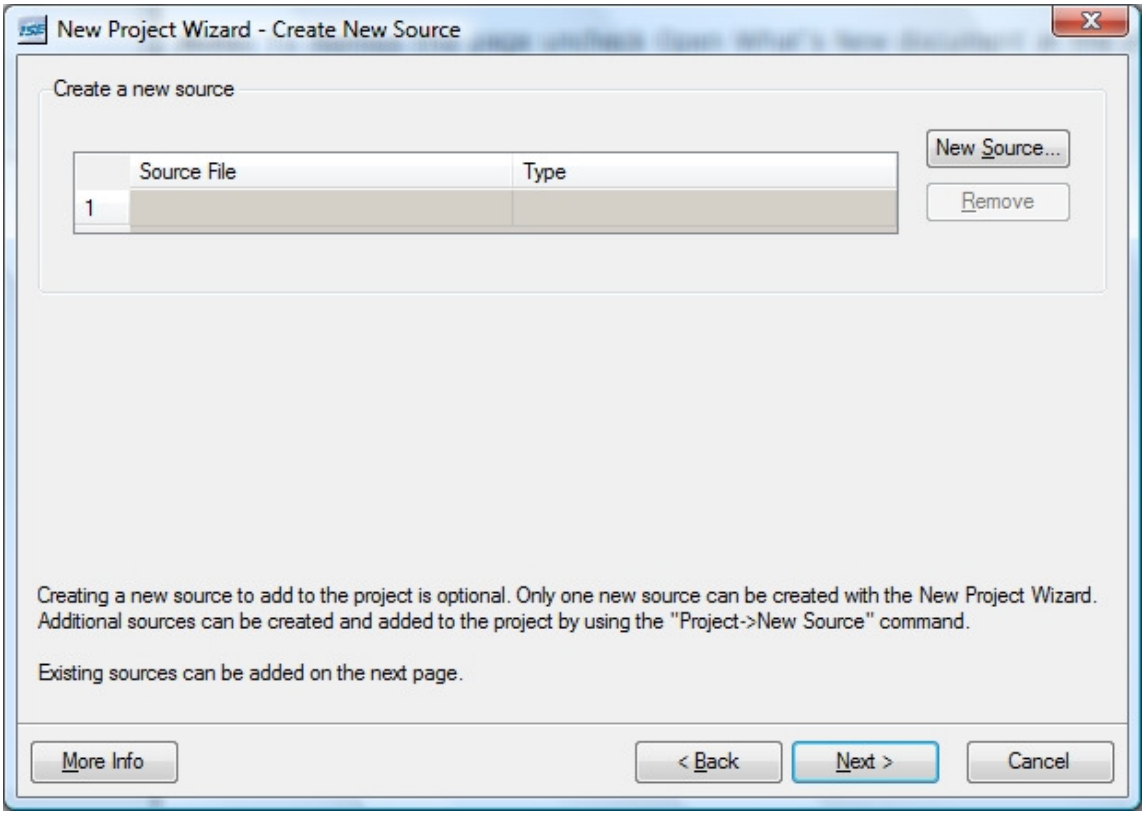


Figura 2.5

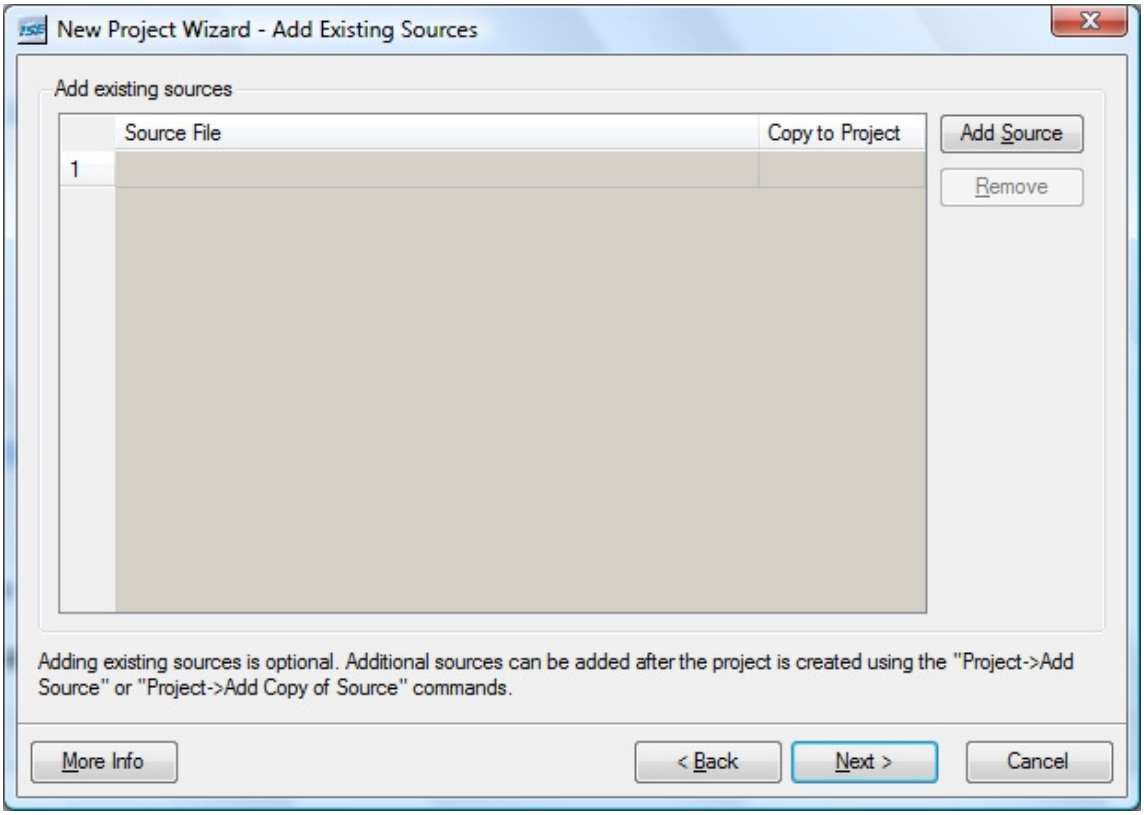


Figura 2.6

- In ultima fereastra din cadrul crearii proiectului va aparea un sumar al setarilor alese anterior.

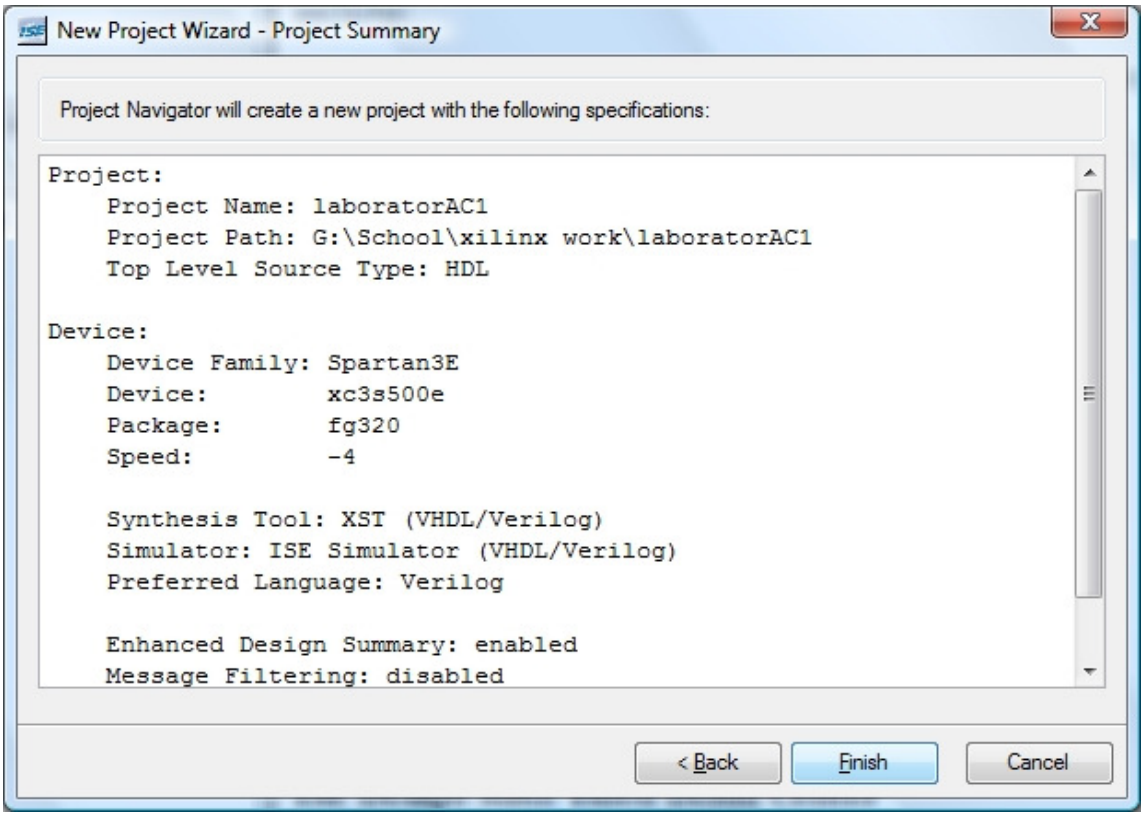


Figura 2.7

## 2.2 Crearea surselor

- Project-> New Source (Figura 3.1)

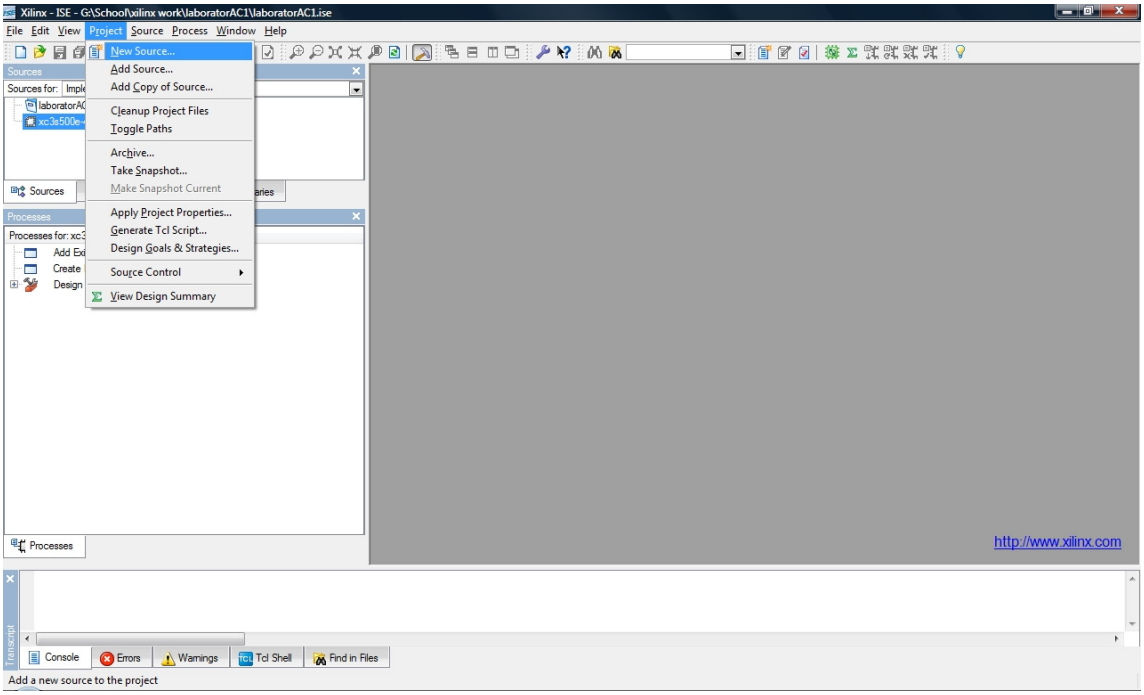


Figura 3.1

- Pentru noul fisier sursa se alege ca tip „Verilog Module" iar numele fisierului se va completa cu „two_input_xor" (Figura 3.2)

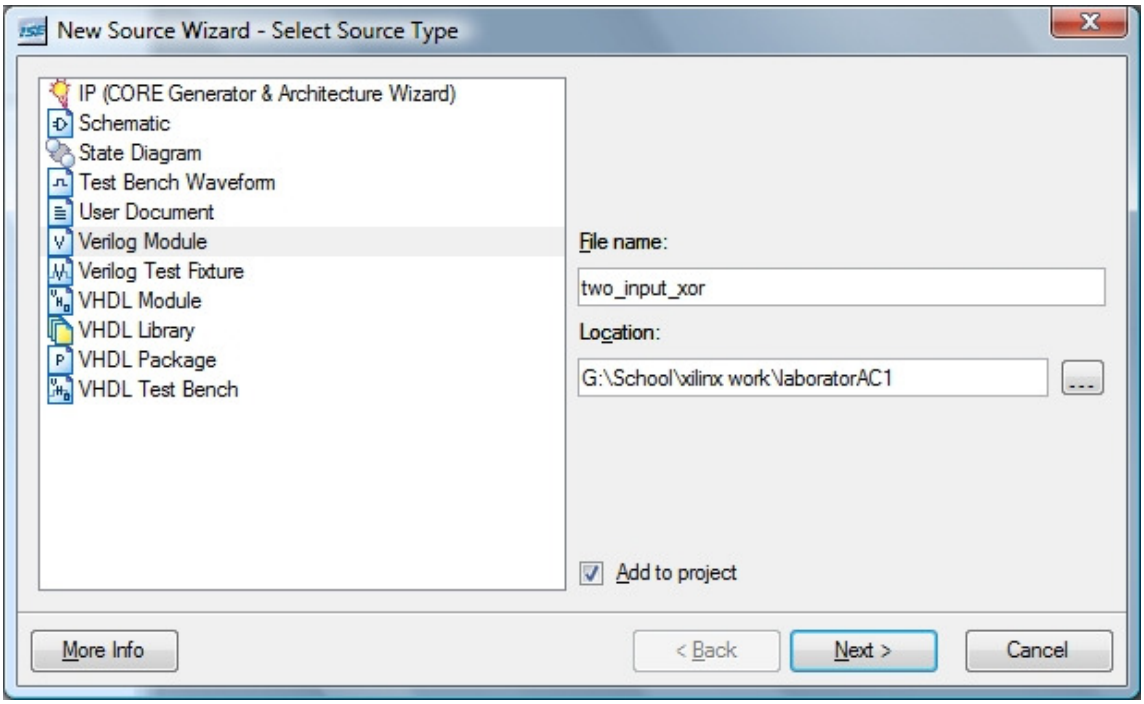


Figura 3.2

- In Figura 3.3 se ilustreaza modul de completare a porturilor de intrare / iesire pentru modulul Verilog nou creat.

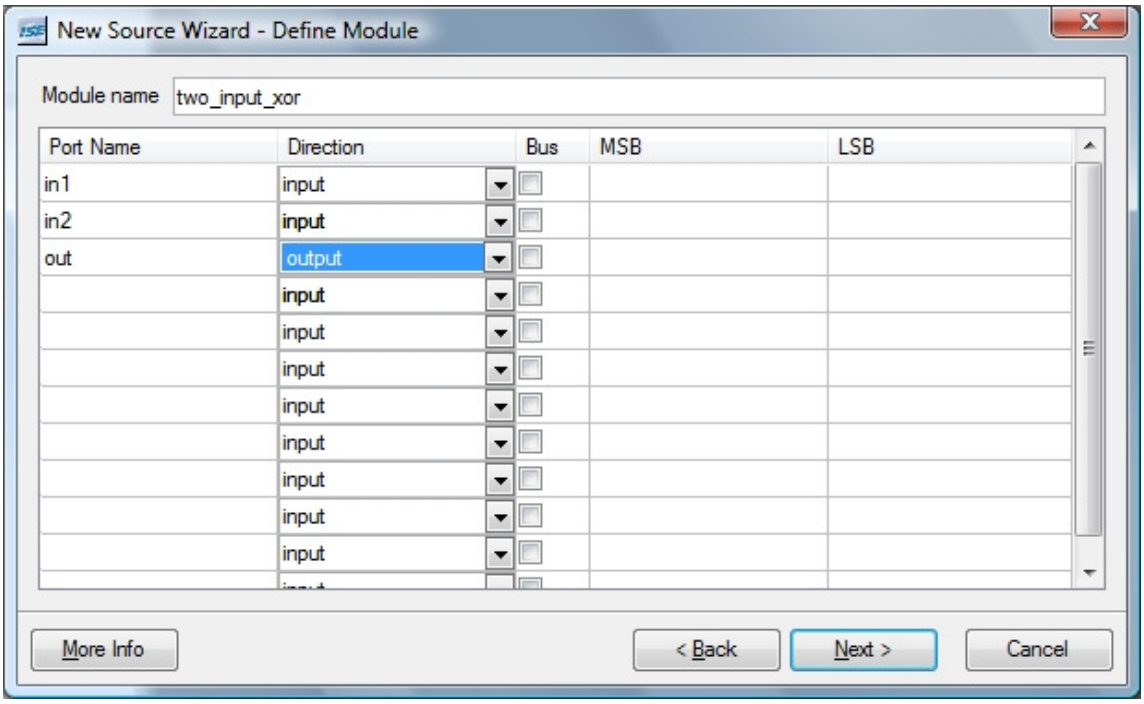


Figura 3.3

- La finalizarea crearii fisierului sursa, va aparea sumarul setarilor (Figura 3.4)

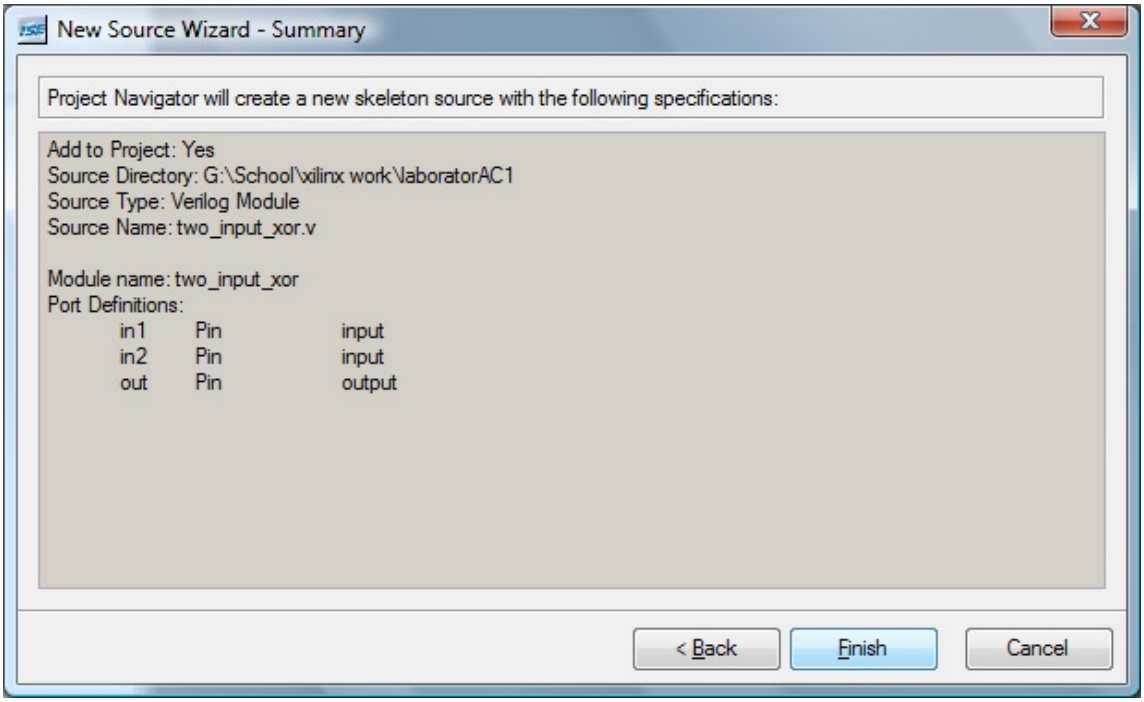


Figura 3.4

- Corpul modulului din fisierul sursa genereat, se va completa cu urmatoarea linie:

```
assign out = in1 ^ in2;
```

ca in Figura 3.5.

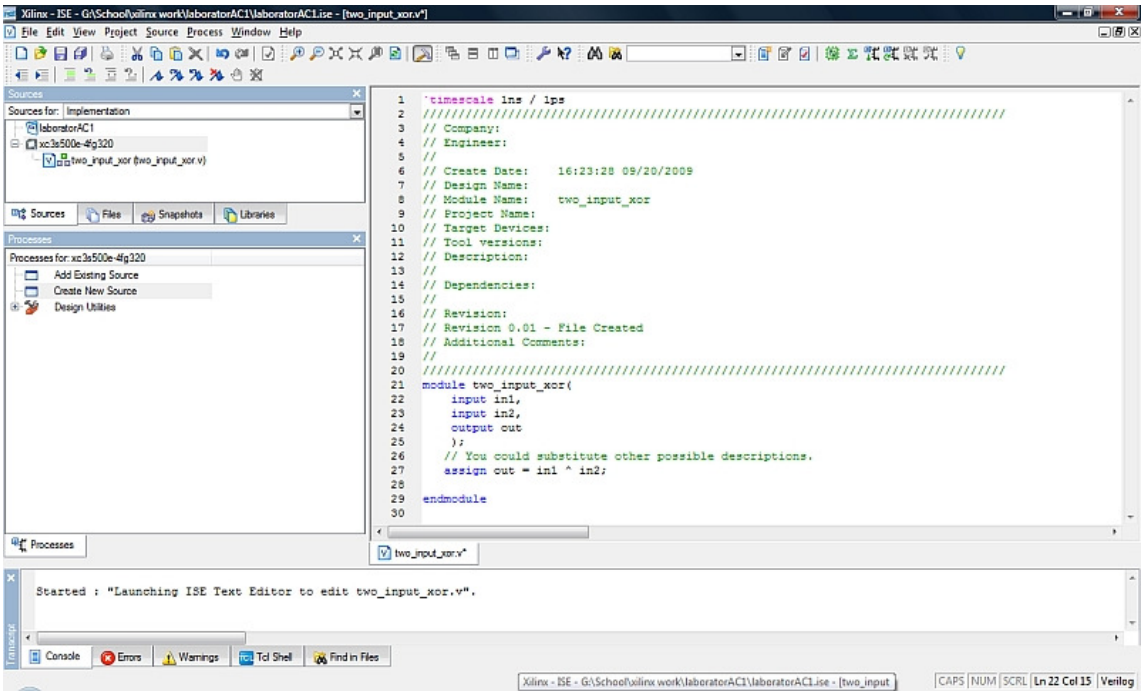

Figura 3.5

- Pentru verificarea sintaxei se selecteaza sursa „two_input_xor.v” din cardul ferestrei „Sources”, se expandeaza „Sysnthesize - XST” apoi se face dublu-click pe „Check Syntax”:

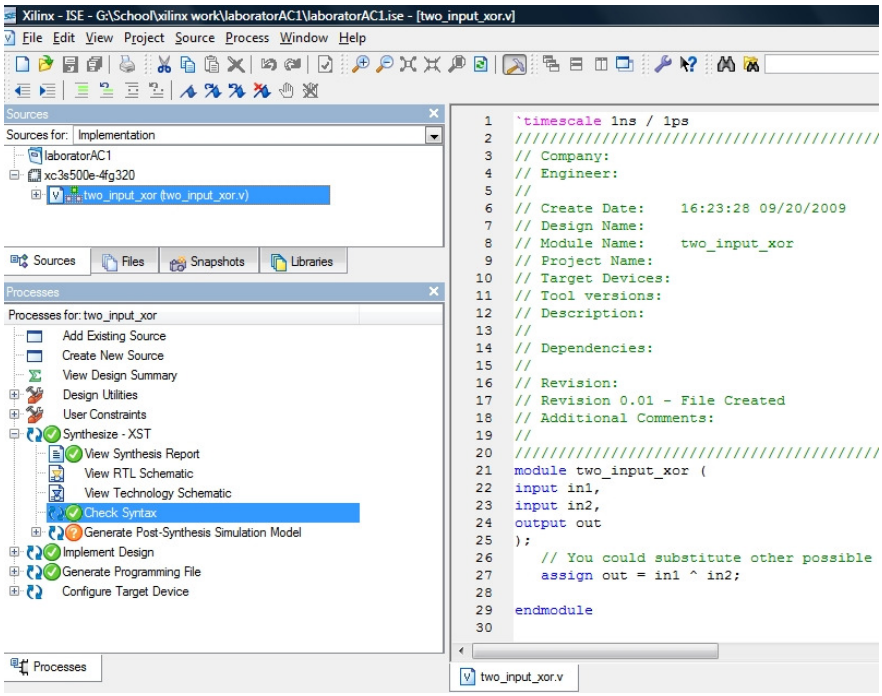

Figura 3.6

## 2.3 Simularea functionalitatii

- Pentru a simula functionarea modulului este necesara creerea unei noi surse (dublu-click „Create New Source” Figura 4.1):

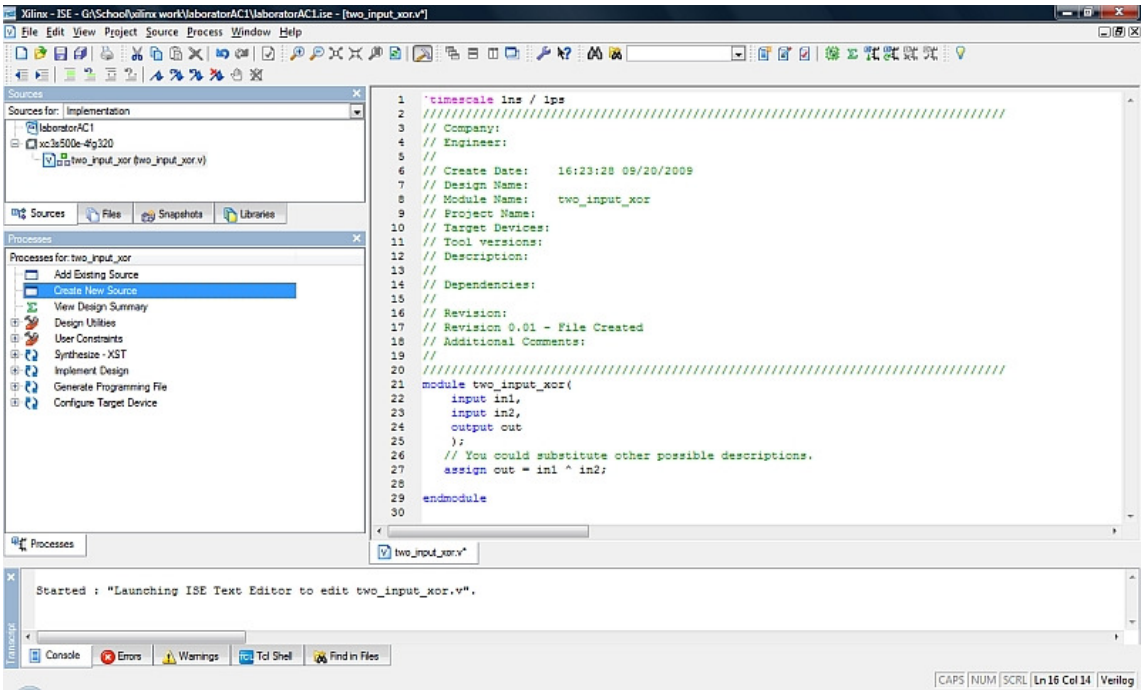


Figura 4.1

- Noua sursa va fi de tipul „Test Bench Waveform” si va purta numele sugestiv „two_input_xor_tbw” ca in Figura 4.2.

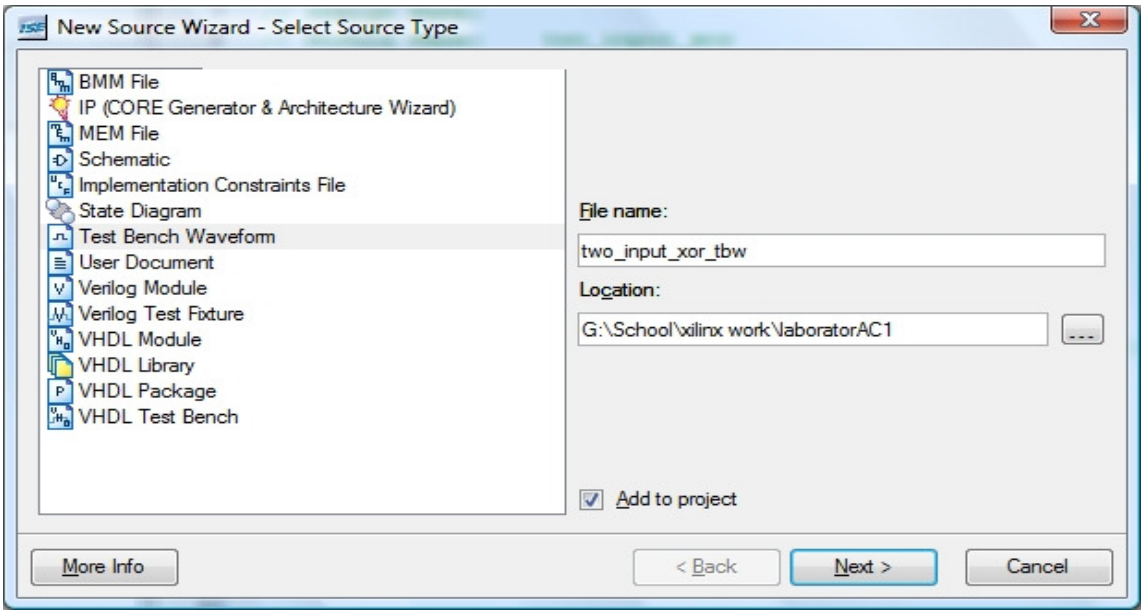


Figura 4.2

- Ulterior se alege modulul caruia i se ataseaza fisierul de simulare (Figura 4.3).

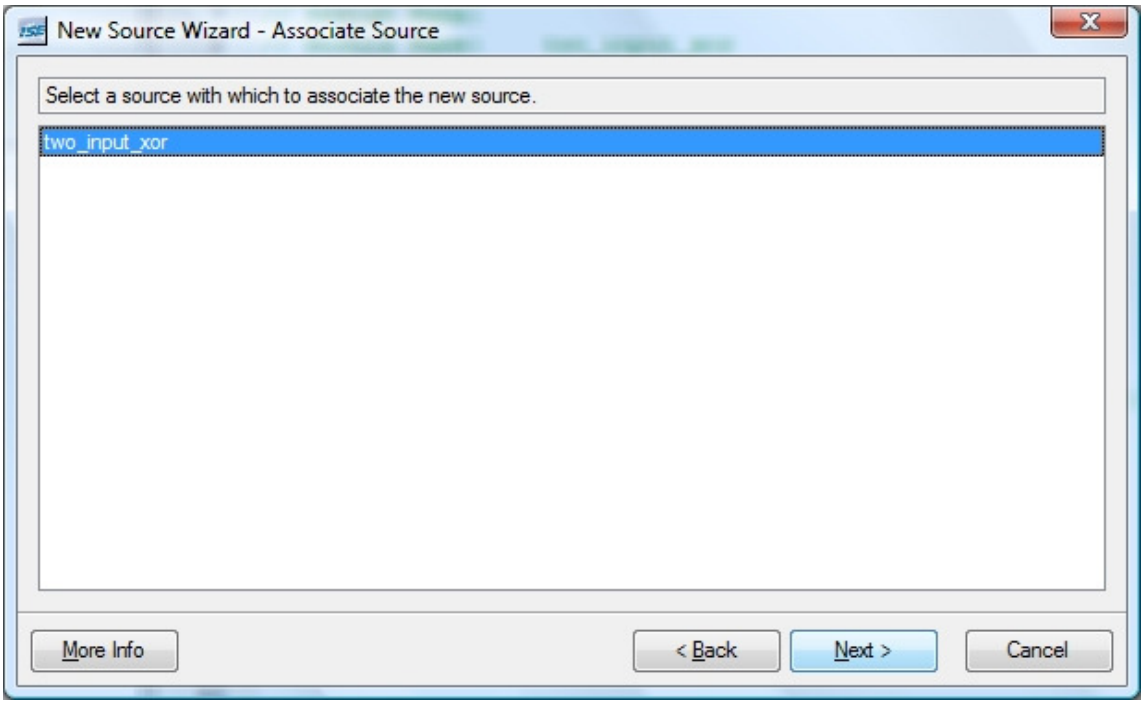


Figura 4.3

- La finalizarea crearii fisierului sursa, va aparea sumarul setarilor (Figura 4.4)

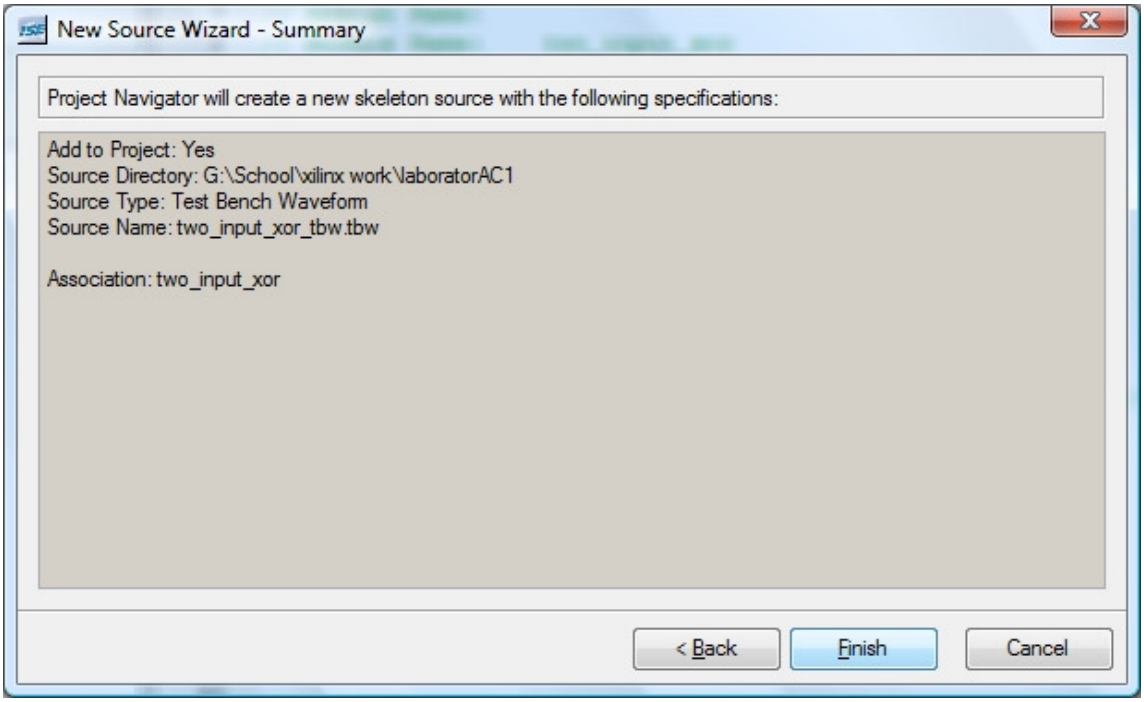


Figura 4.4

- Pentru realizarea simularii comportamentale a modulului trebuie setati parametri de simulare conform figurii 4.5:

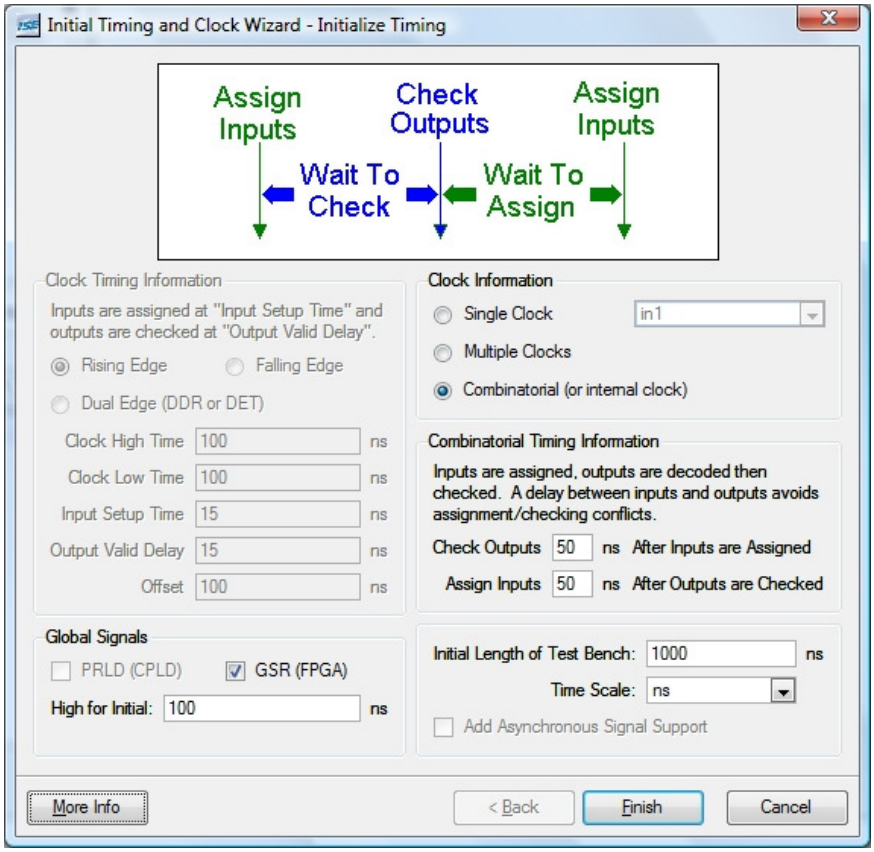


Figura 4.5

- Variabilele de intrare („in1” si „in2”) se seteaza manual ca in Figura 4.6.

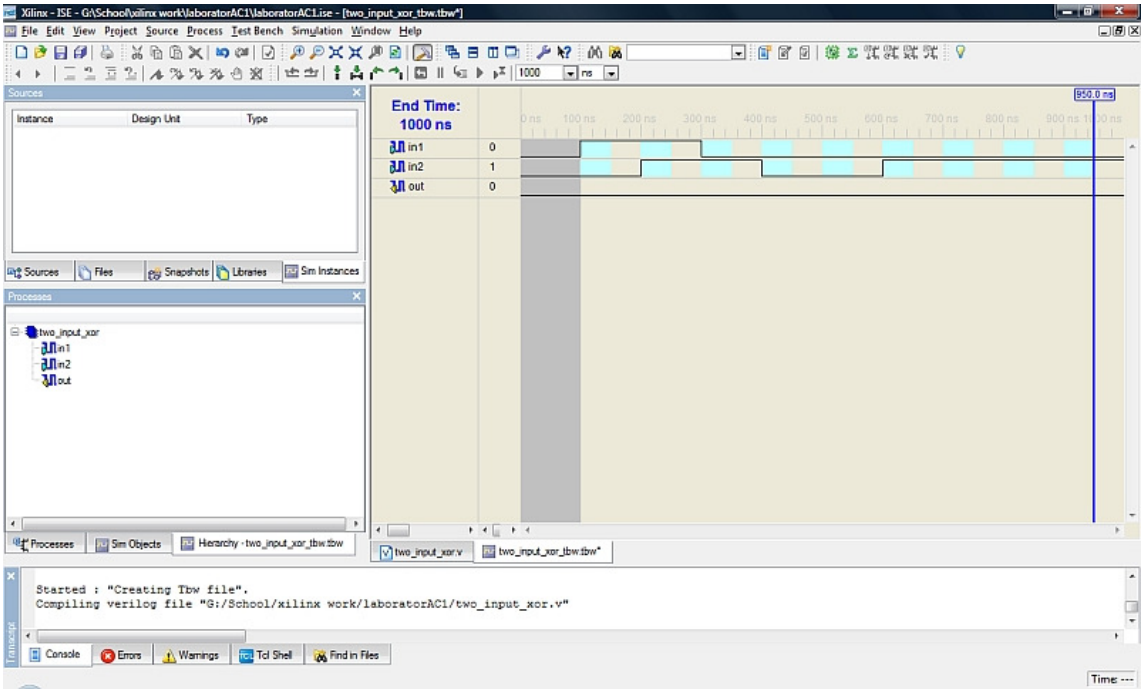


Figura 4.6

- Pentru a terce in modul de simulare se va selecta „Behavioral Simulation” din fereastra „Sources” ca in Figura 4.7.

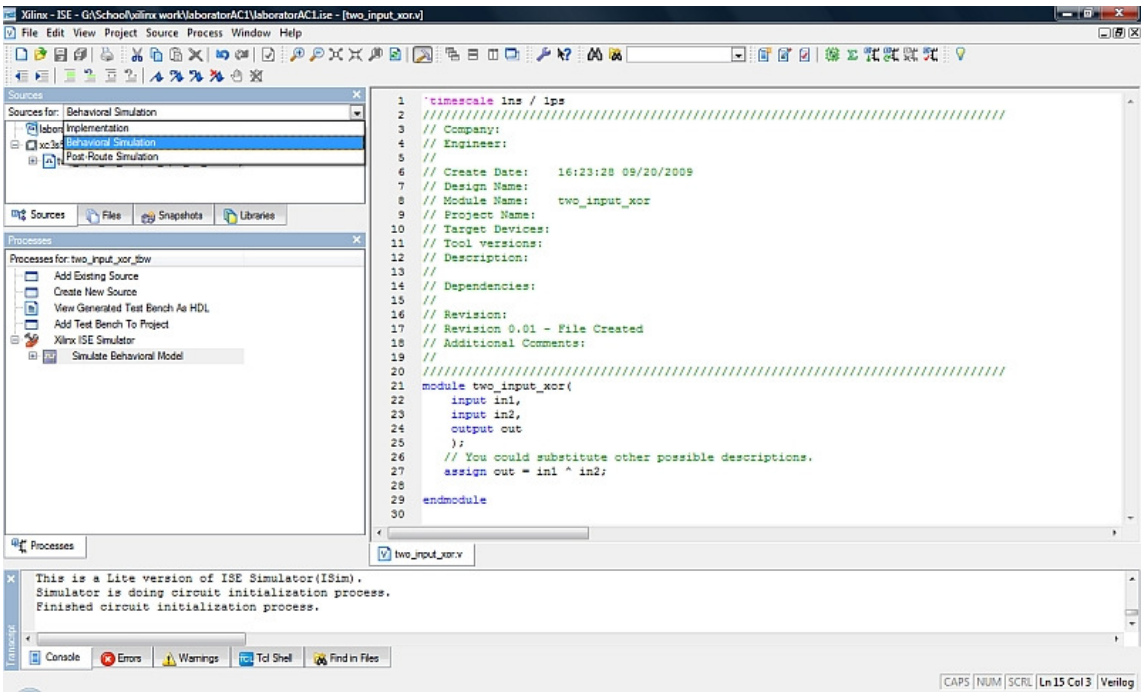


Figura 4.7

- Pornirea simularii se face prin dublu-click pe „Simulate Behavioral Model” (Figura 4.8).

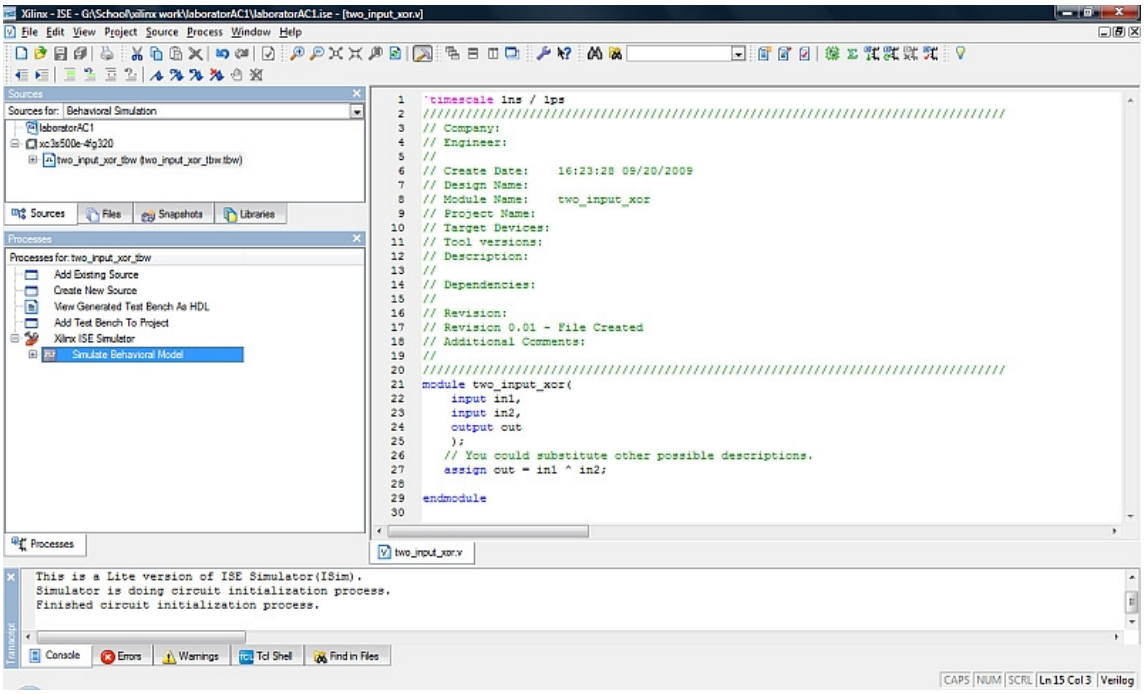


Figura 4.8

- Ca rezultat a simularii se obtin urmatoarele variatii de semnale(Figura 4.9):

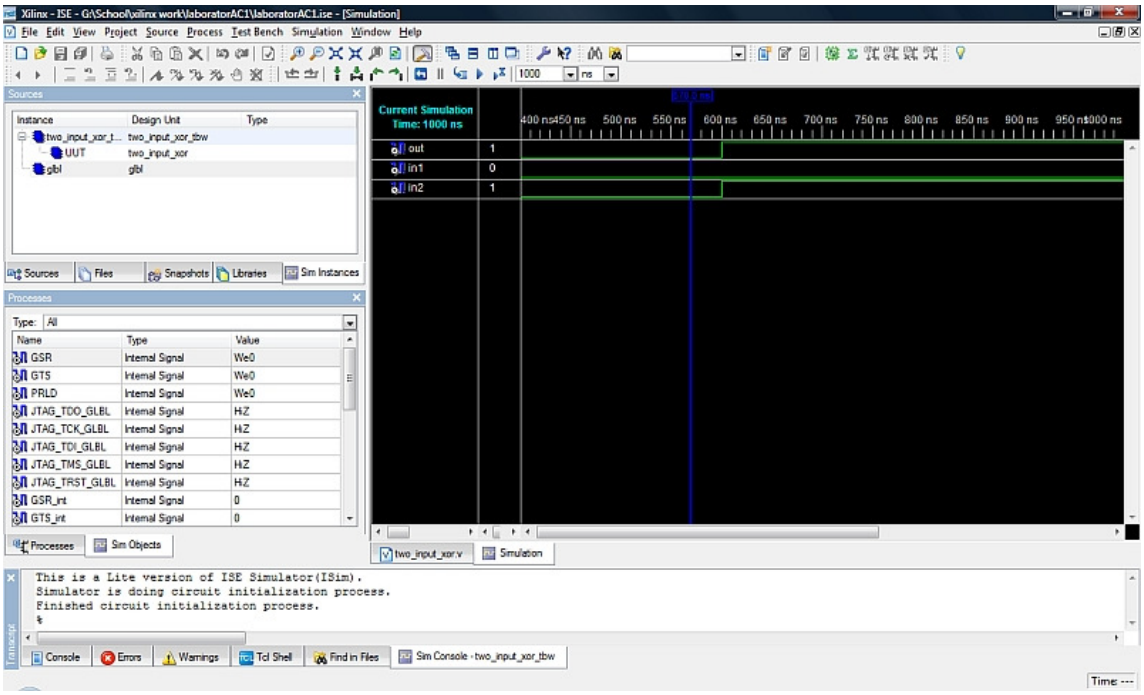


Figura 4.9

## 2.4 Sintetizarea proiectului

- Fiind proiectata descrierea unui circuit in limbajul Verilog – HDL urmatorul pas este de a folosi un mecanism de sintetizare (in acest caz: XST – Xilinx Synthesize Tool) pentru a transforma codul VHDL intr-o retea de porti logice pentru a fi implementata pe FPGA.
- Sintetizarea se face prin dublu – click pe „Synthesize - XST" ca in Figura 5.1.

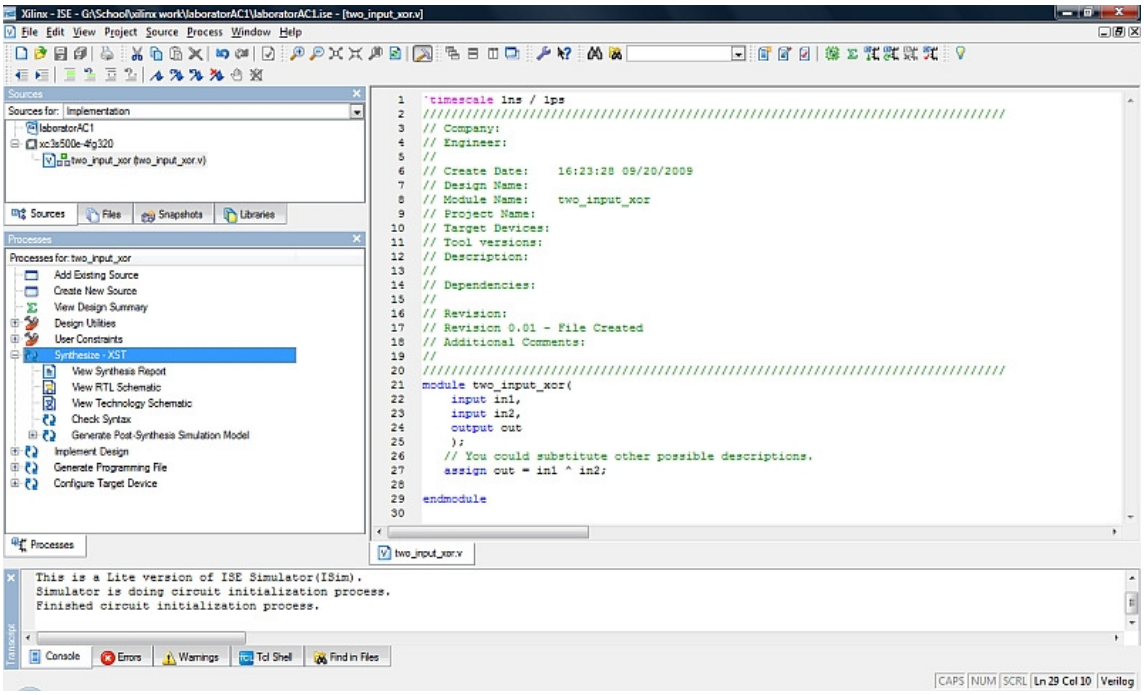


Figura 5.1

## 2.5 Programarea FPGA-ului

- Pentru a programa FPGA-ul cu functionalitatea descrisa anterior, sunt necesare definirea unor constrangeri specifice tipului placii. Pentru acestea se ceaza o noua sursa de tin „UCF” (User Constraints File) (Figura 6.1).

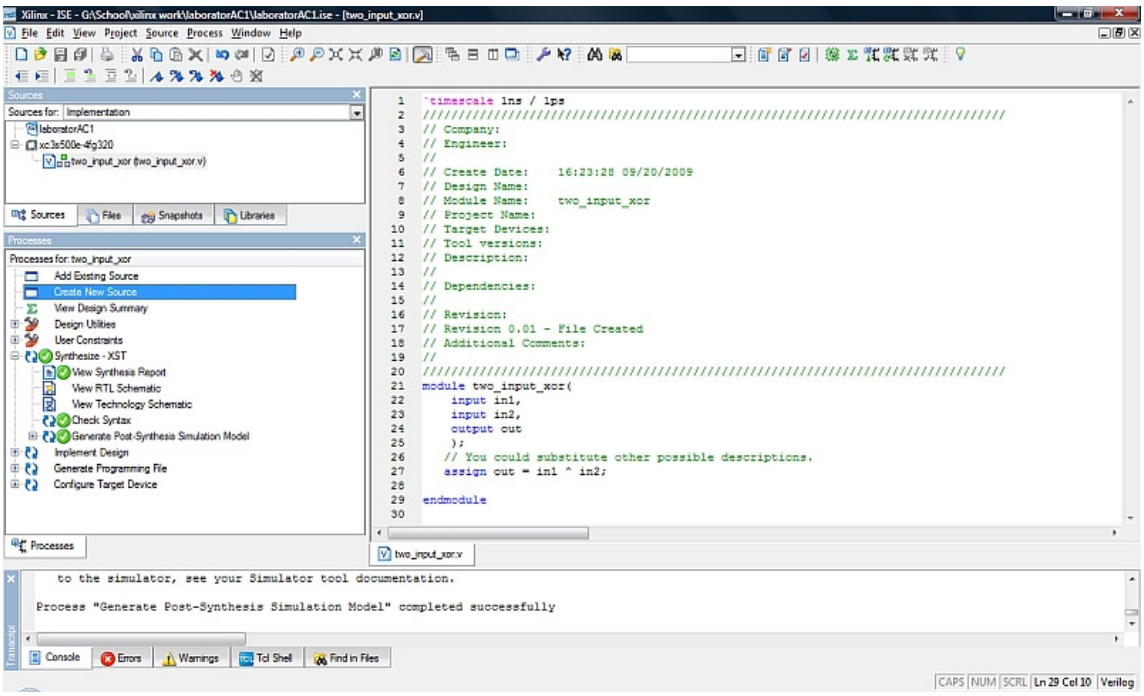


Figura 6.1

- La crearea fisierului se va selecta ca tip „Implementation Constraints File” si se va denumi „two_input_xor_ucf” ca in figura 6.2.

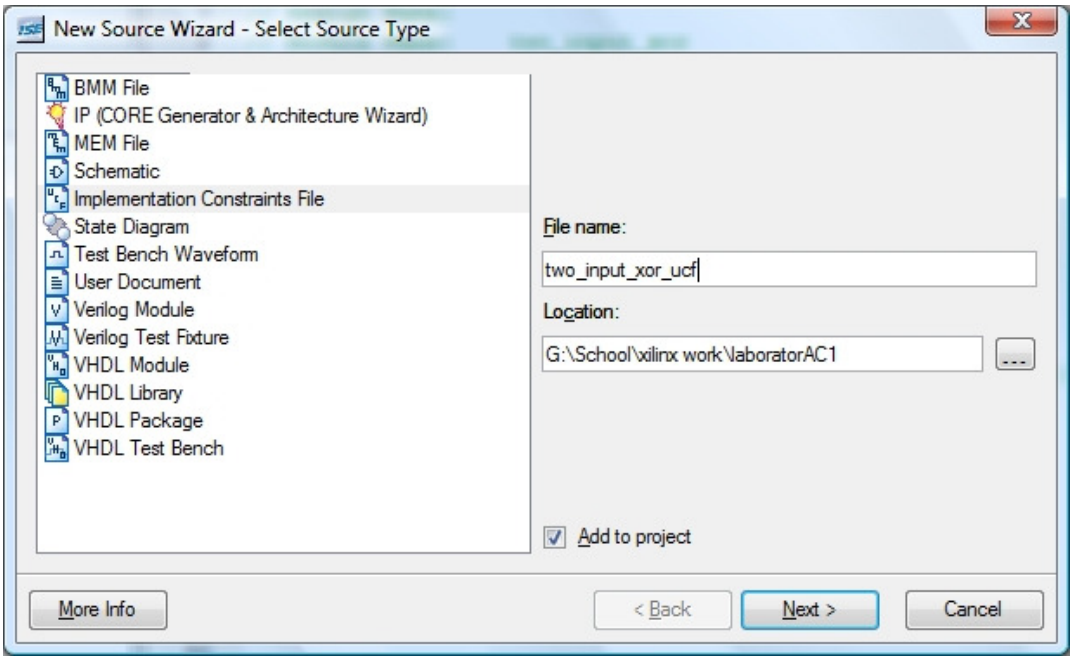


Figura 6.2

- Se va obtine un sumar al setarilor realizate (Figura 6.3).

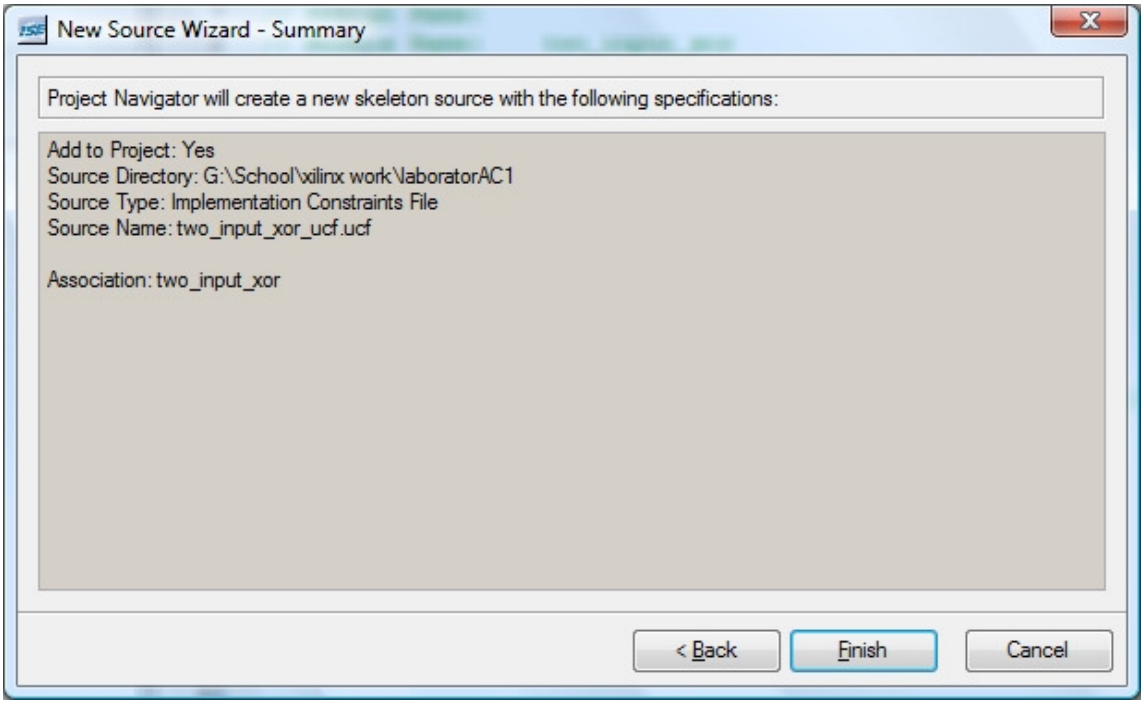


Figura 6.3

- Folosind sectiunea „User Constraints" din fereastra „Processes" se poate edita fisierul de constrangeri nou creat. In acest caz se urmareste atribuirea de pini fizici porturilor de intrare / iesire ai modulului proiectat. Pentru a deschide editorul „PACE" se face dublu-click pe „Floorplan Area/ IO/ Logic – Port – Synthesis" (Figura 6.4).

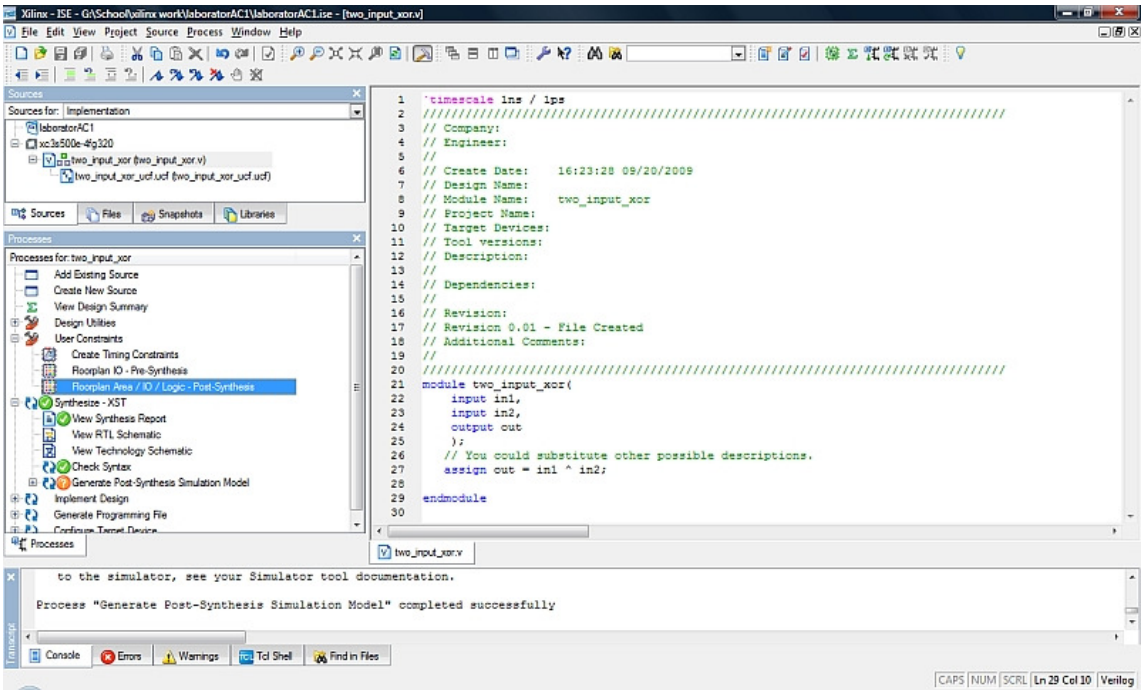


Figura 6.4

- Se selecteaza „Package View”si se vor atribui urmatorarele switch-uri pentru cele doua intrari „L13”(in1) respectiv „L14”(in2) si LED-ul pentru iesire „F12”(out) conform figurii 6.5.

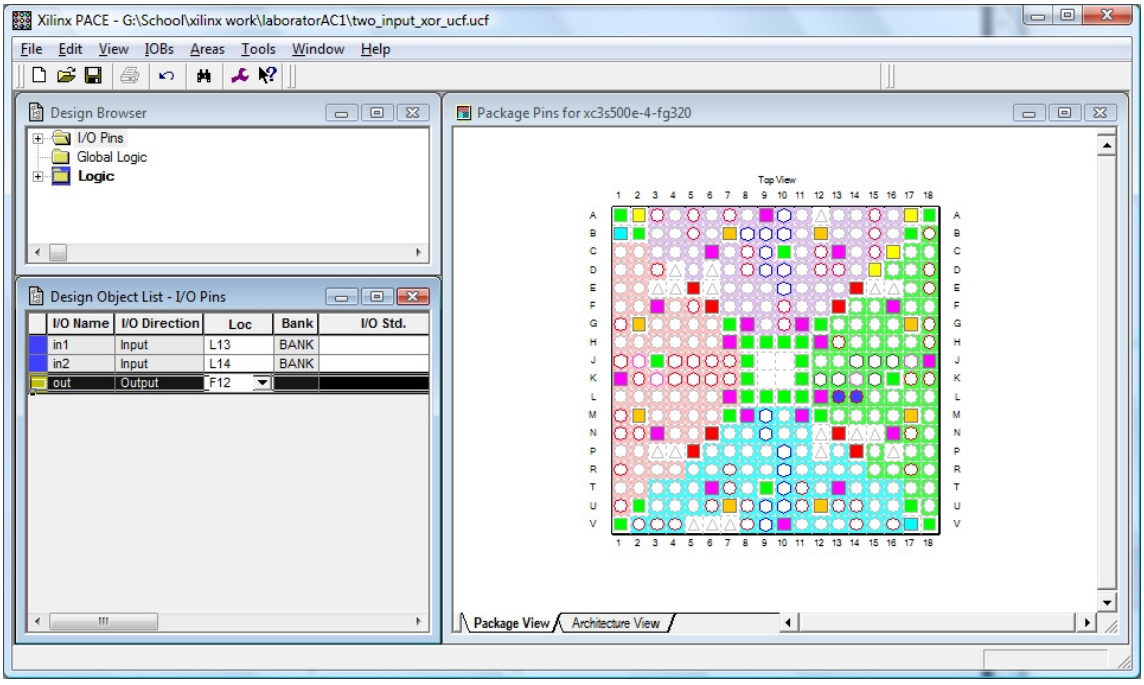


Figura 6.5

- Dupa editarea fisierului de constrangeri se trece la implementarea circuitului. Pentru aceasta se selecteaza sursa „two_input_xor.v” din fereastra „Sources” si se face dublu-click pe „Implement Design” din fereastra „Processes” (Figura 6.6)

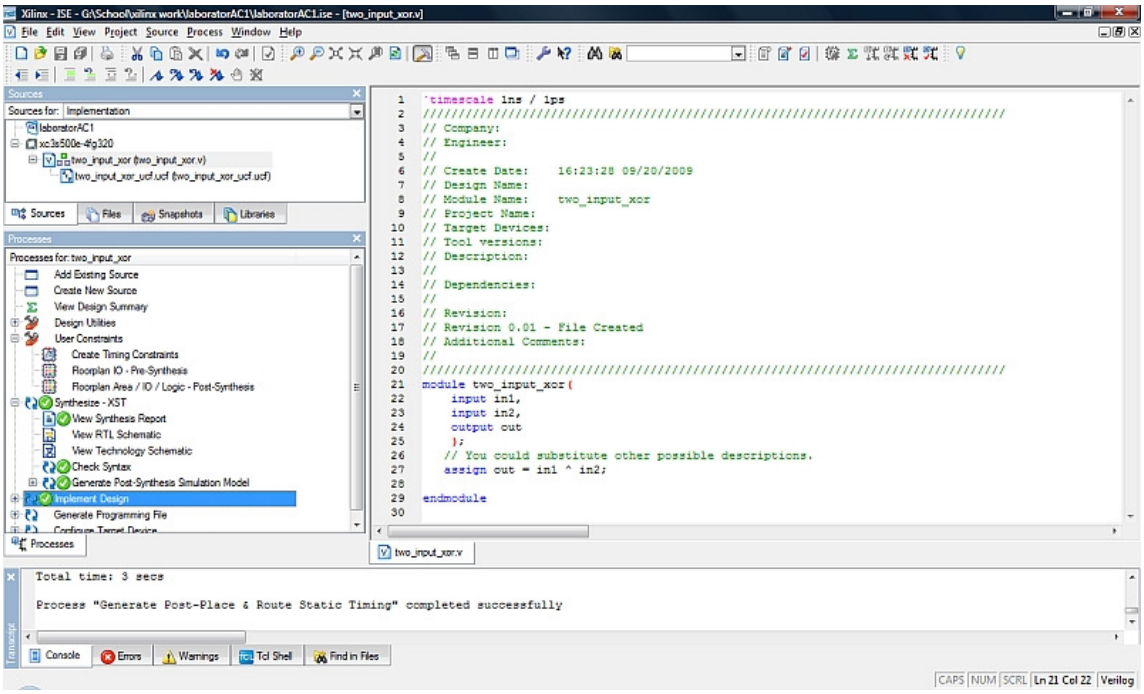


Figura 6.6

- Ulterior se face dublu-click pe „Generate Programming File” din fereastra „Processes” (Figura 6.7).

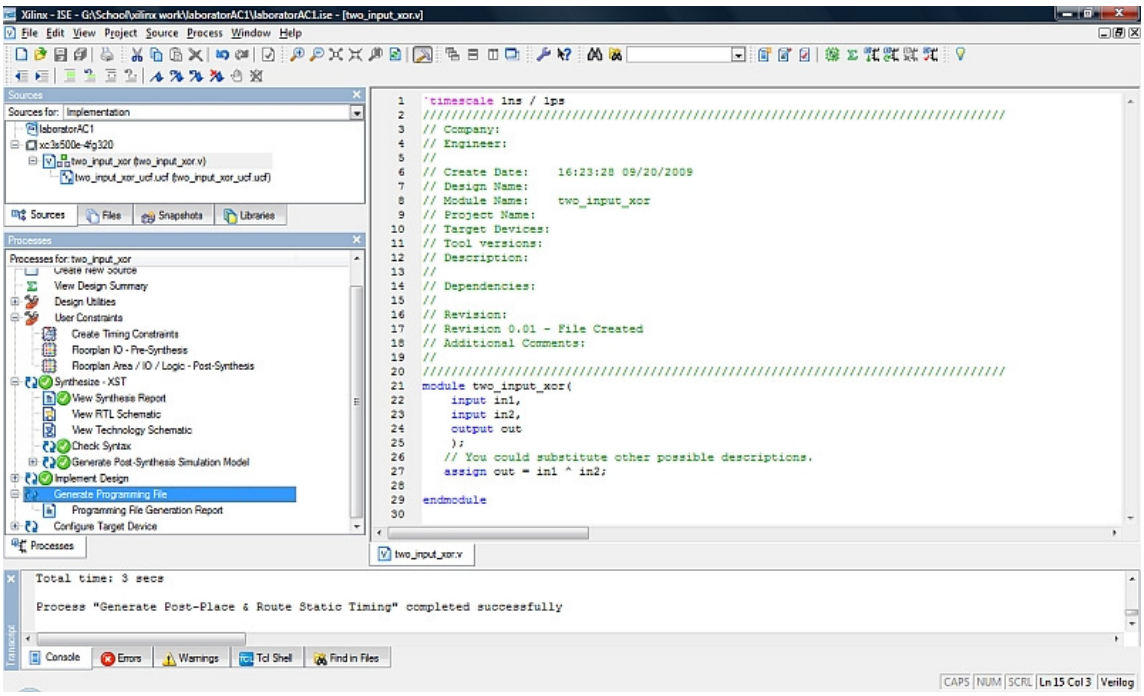


Figura 6.7

- In acest moment este necesara alimentarea placii SPARTAN 3E si conectarea acesteia la PC. Dupa pornirea placii si recunoasterea ei de catre PC, se face dublu-click pe „Manage Configuration Project (iMPACT)” (Figura 6.8).

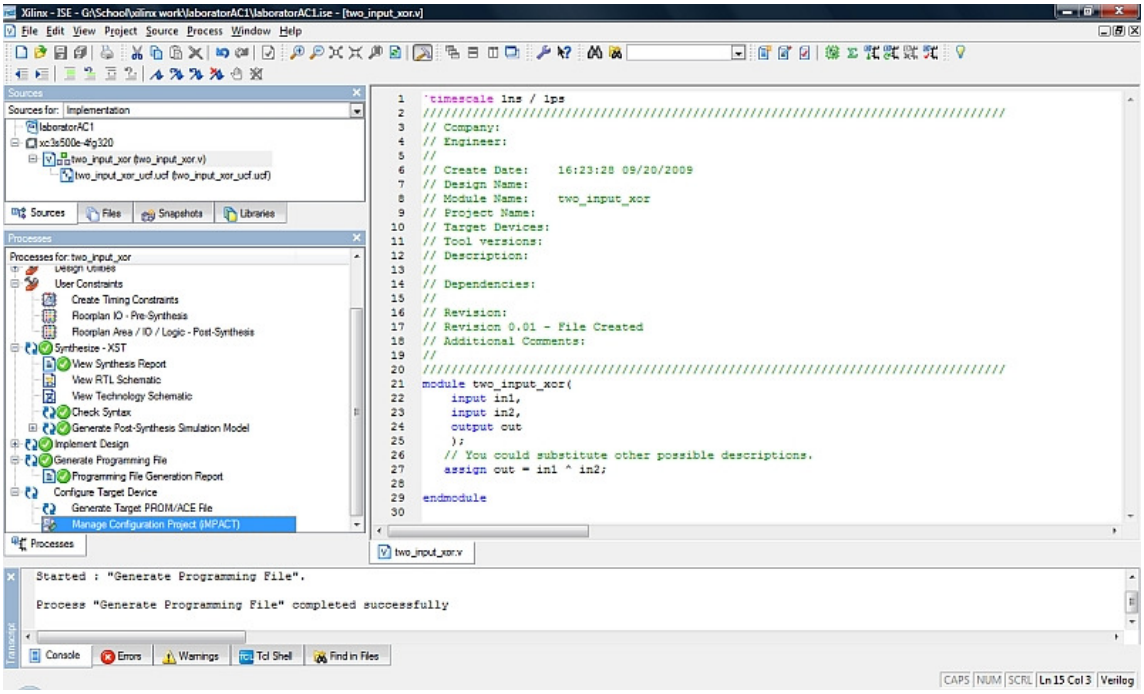


Figura 6.8

- Pentru fereastra urmatoare se vor pastra setarile default (Figura 6.9).

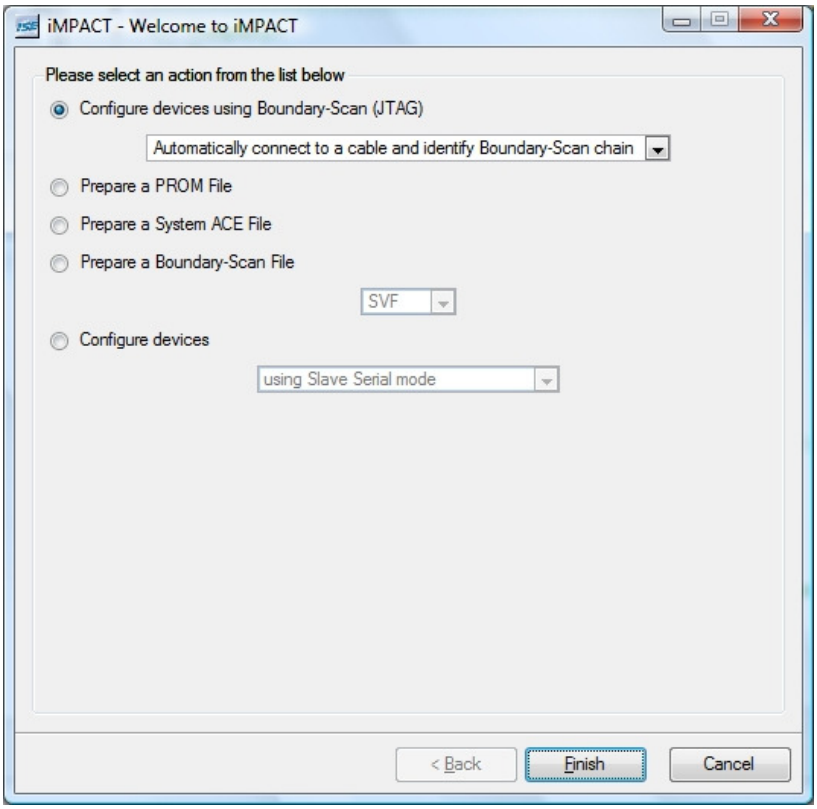


Figura 6.9

- FPGA-ului XC3S500E i se va atribui fisierul „two_input_xor.bit” (Figura 6.10).

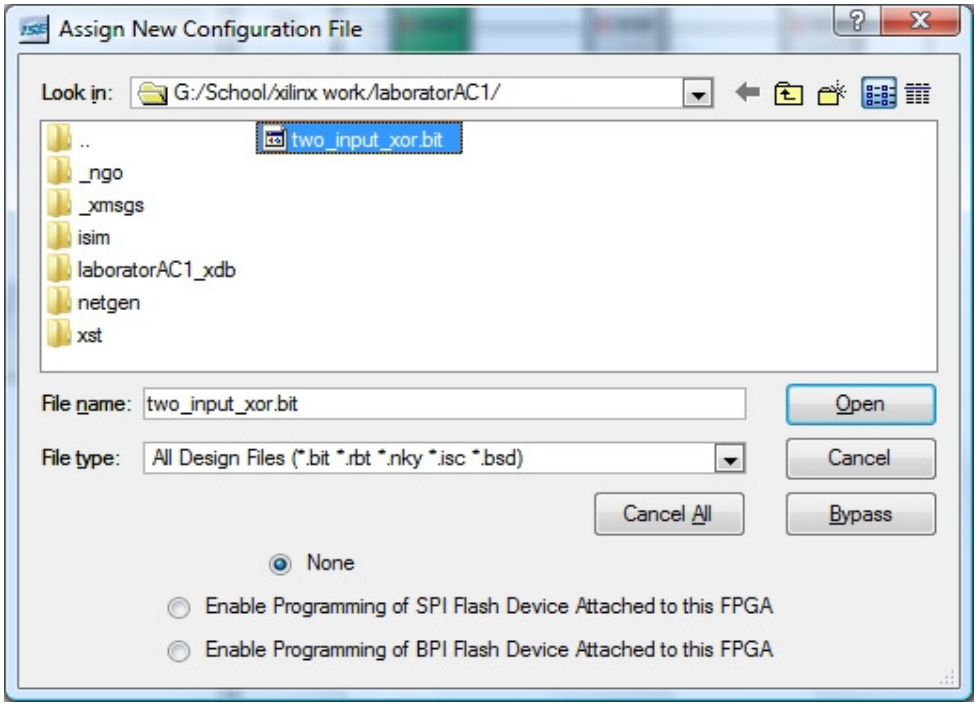


Figura 6.10

- Celui de-al doilea device i se va atribui fisierul „xcf04s.bsd” (Figura 6.11)

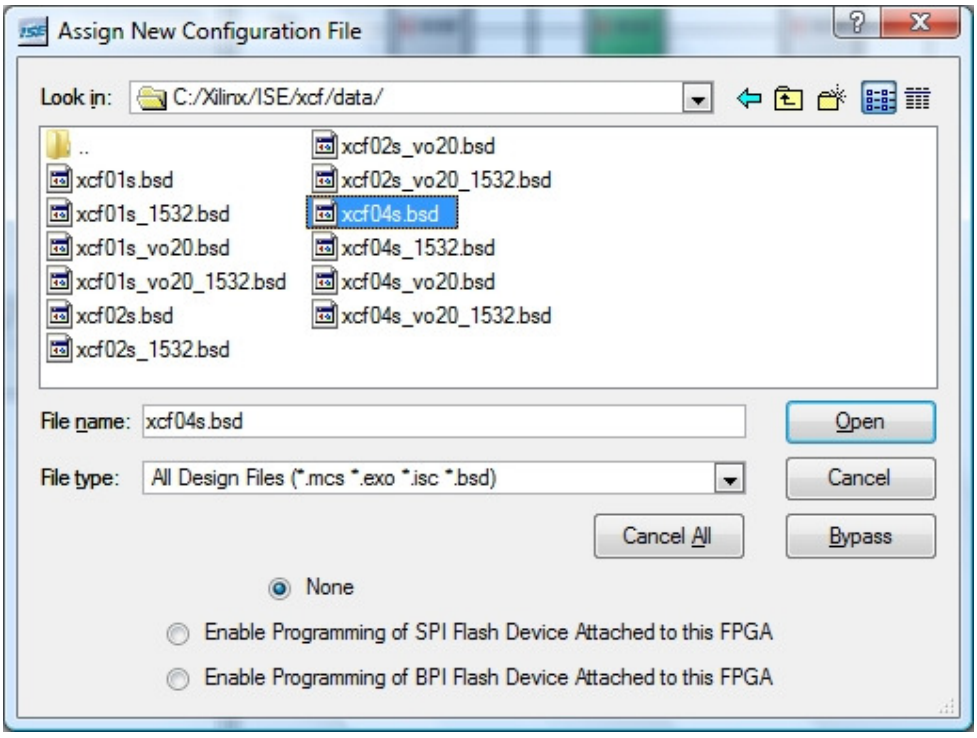


Figura 6.11

- Ultimului device i se va atribiu fisierul „xc2c64a.bsd” (Figura 6.12)

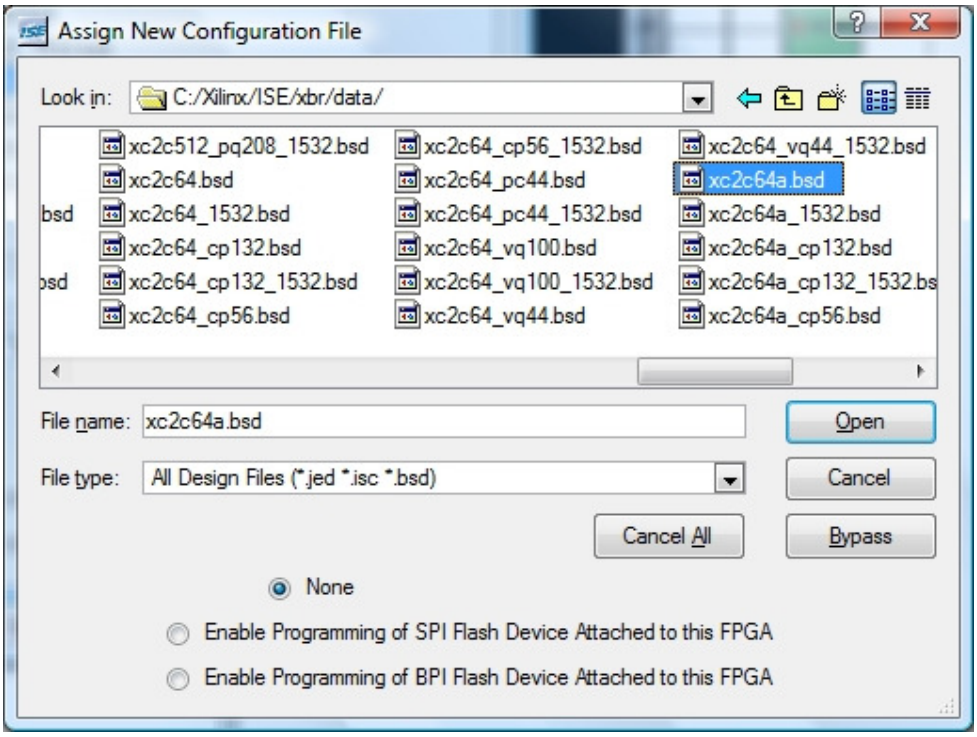


Figura 6.12

- Pentru urmatoarea fereastra se vor pastra setarile default (Figura 6.13).

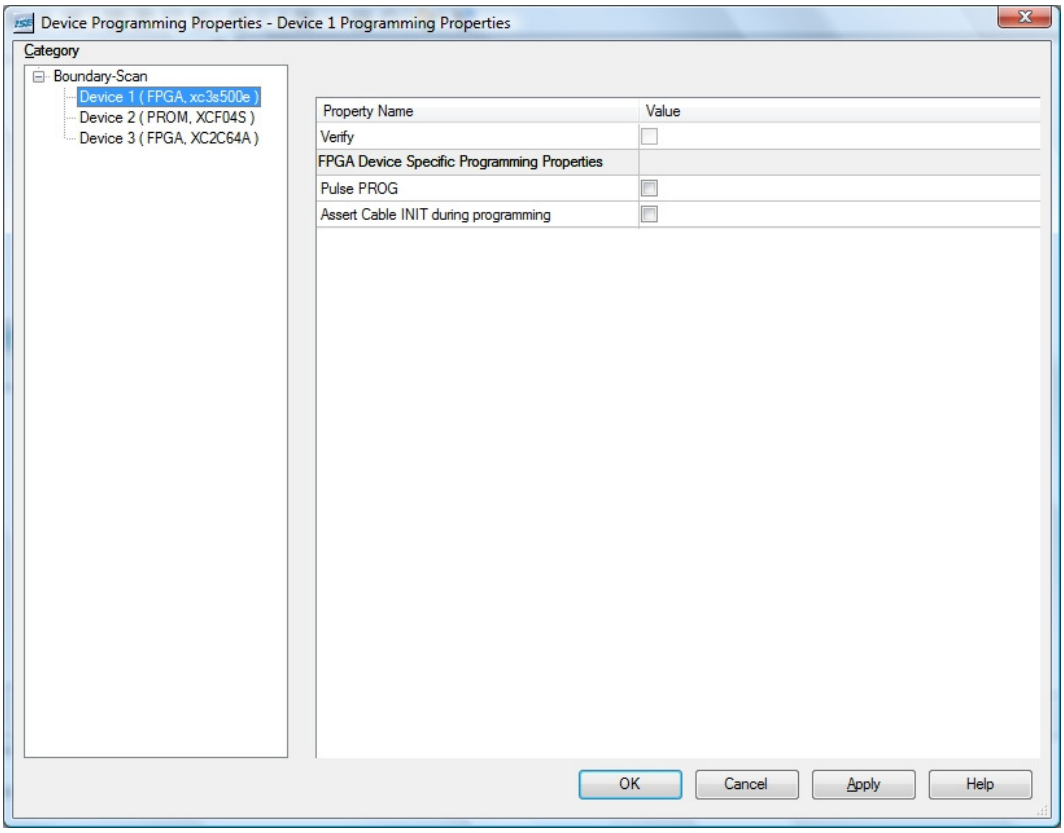


Figura 6.13

- Programarea efectiva a FPGA-ului se realizeaza prin selectarea primului dispozitiv, apoi click dreapta -> „Program” (Figura 6.14)

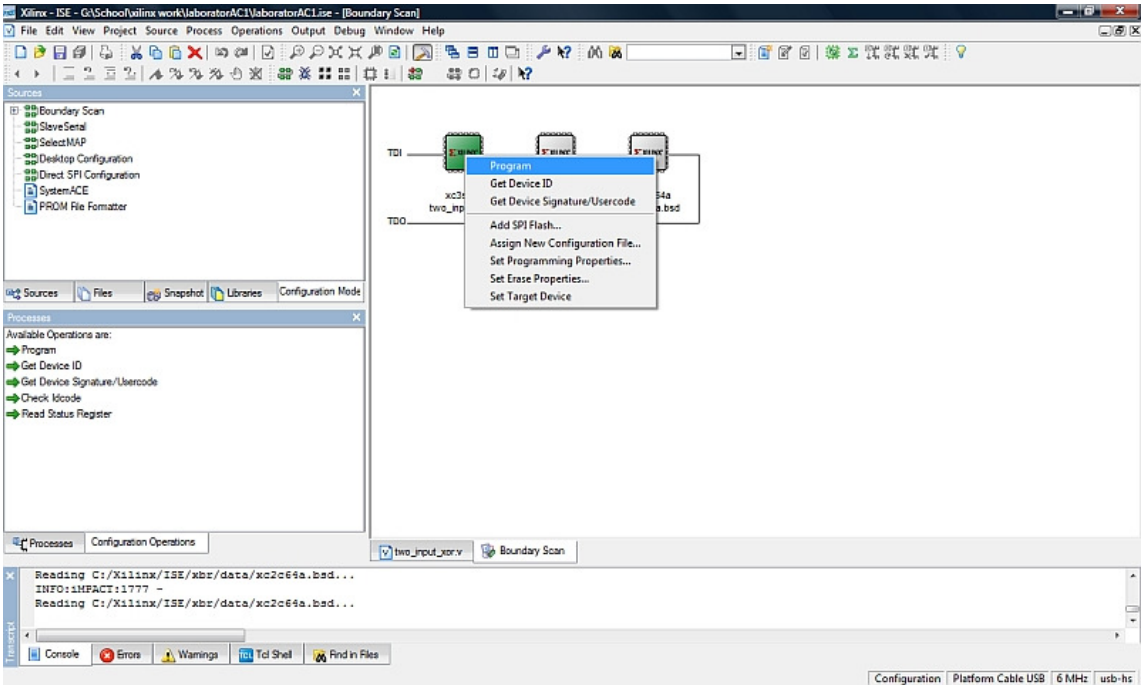


Figura 6.14

- In conditiile in care toti pasii au fost urmati corect se va obtine „Program Succeeded” (Figura 6.15).

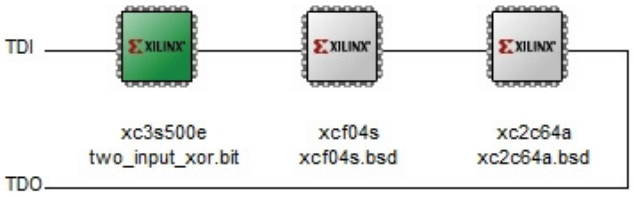



Figura 6.15

# 3. Testarea Divizorului pe placa

Aplicatia va fi testata pe placa prin aprinderea a doua leduri: un led va fi legat la intarea modului realizat la tema iar cel de-al doilea led va fi legat la iesirea modulului. Intrarea modulului este ceasul de pe placa de 50MHz, divizat la 2Hz pentru a putea observa „licaririle" ledurilor. Pentru atribuirea pinilor se va utiliza „Spartan-3E FPGA Starter Kit Board User Guide" (ug230.pdf).

Se vor folosi urmatoarele surse Verilog (pentru claritate in fisiere separate) pentru a crea un proiect Xilinx cu functionalitatea descrisa anterior.

```
module top_module(CLK_50MHz, CLK_2Hz, CLK_1pe2Hz_1pe4);

	input CLK_50MHz;
	output CLK_2Hz;
	output CLK_1pe2Hz_1pe4;

	super_divizor_clk div(CLK_50MHz, CLK_2Hz);
	generator_impulsuri gen(CLK_2Hz, CLK_1pe2Hz_1pe4);

endmodule

module super_divizor_clk(CLK_50MHz, CLK_2Hz);
	//rezulta un ceas cu perioada 0.5 secunde

	input CLK_50MHz;
	output CLK_2Hz;
	reg CLK_2Hz;

	reg [23:0] contor;

	initial
	begin
		CLK_2Hz <= 0;
		contor <= 1;
	end

	always @(posedge CLK_50MHz)
	begin
		if (contor == 12500000)
		begin
			CLK_2Hz <= ~CLK_2Hz;
			contor <= 1;
		end
		else
			contor <= contor + 1;
	end

endmodule
```

```
module generator_impulsuri(clk, out);

	input clk;
	output out;
	reg out;
	reg [1:0] contor;

	initial
	begin
		out <= 0;
		contor <= 0;
	end

	always @(posedge clk)
	begin
		if (contor == 0)
			out <= 1;
		else
			out <= 0;
		contor <= contor + 1;
	end

endmodule
```