Configurarea structurilor hardware reprogramabile de tip FPGA cu XILINX ISE

I. SCOPUL LUCRĂRII

În această lucrare este prezentată metodologia relativă la implementarea programelor scrise prin limbajul VHDL, pe structura reprogramabilă FPGA de tip SPARTAN 3, folosind pachetul software de dezvoltare XILINX ISE. Pe scurt, sunt abordate următoarele etape: introducerea codului VHDL, simularea şi vizualizarea rezultatelor obținute, sinteza şi implementarea surselor, iar în final programarea structurii reconfigurabile.

II. INTRODUCERE TEORETICĂ

Mediul software de proiectare XILINX ISE (Integrated Software Environment) este utilizat la realizarea şi implementarea completă a unui proiect pentru structurile programabile de tip XILINX. Componenta software ISE Project Navigator facilitează crearea proiectului, organizându-l pe următoarele etape:

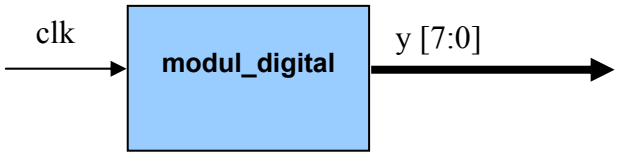
- *Descriere proiect.* Programatorul are posibilitatea să descrie proiectul prin introducerea de coduri sursă HDL (Hardware Description Language) pentru limbajele VHDL, Verilog, Abel sau utilizând schematice şi diagrame cu stări finite;
- *Sinteza.* În cadrul acestei etape fişierele de tip VHDL, Verilog sau schematice sunt transformate în fişiere de tip *netlist* care sunt acceptate ca fişiere de intrare la etapa de implementare;
- *Implementarea.* După sinteză, la implementare, proiectul este adaptat şi transformat din forma logică digitală în forma tehnologică implementabilă pe structura reconfigurabilă aleasă;
- *Verificarea.* Poate fi realizată în toate etapele de implementare ale proiectului. Utilizarea componentelor software de simulare conduce la verificarea completă a funcționalității proiectului sau a unor porțiuni de proiect. De asemenea, pot fi realizate şi verificări directe pe circuit, după programarea acestuia;
- *Configurarea.* După generarea fişierelor de programare (bitstream file) proiectantul are posibilitatea programării circuitului reconfigurabil. În timpul procesului de configurare sunt programate interconexiunile structurii FPGA alese.

În vederea exemplificării modului de implementare a unui modul digital pe o structură de tip FPGA, se prezintă următorul model.

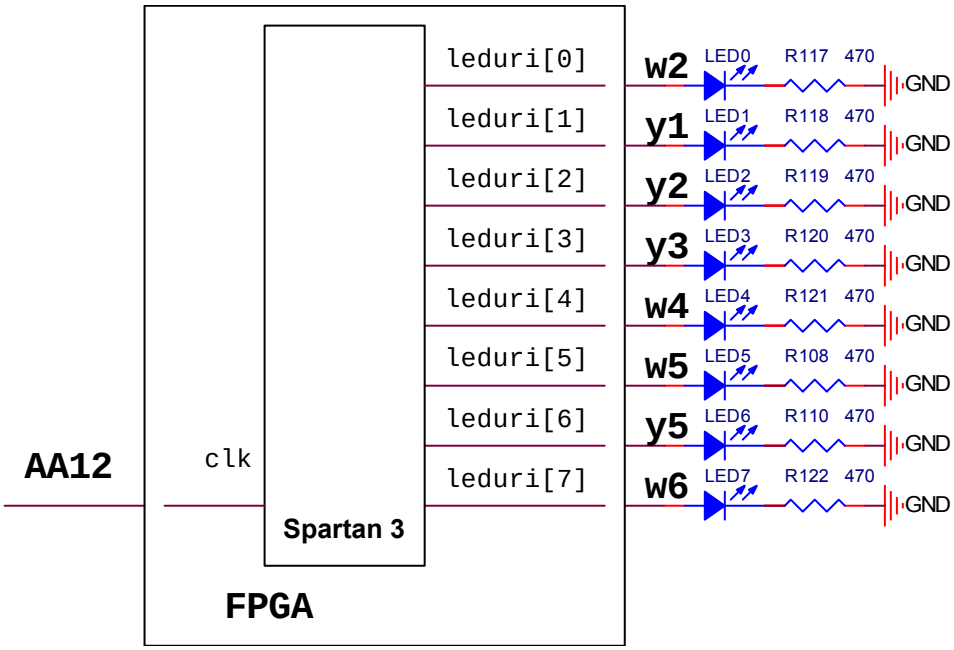
1. Enunțul problemei

Să se descrie etapele de implementare ale unui modul digital numărător binar pe 4 biți, în limbajul VHDL, pe sistemul reconfigurabil cu circuitul FPGA SPARTAN3 din laborator.

Porturile de intrare/ieşire ale modulului digital sunt prezentate în figura următoare:



Verificarea modulului digital va fi realizată folosind schema electrică din figura de mai jos. Această schemă este doar o mică parte din schema electrică a sistemului reconfigurabil de dezvoltare SPARTAN 3:



Notă: Se va consulta schema electrică completă a sistemului de dezvoltare şi se vor identifica pe aceasta elementele de circuit.

Pinii de interconectare ai structurii FPGA cu modulul digital descris în VHDL sunt marcaţi cu bold (**Y6**, **Y17**, **C6**, **B8**, **E7** şi **C5**).

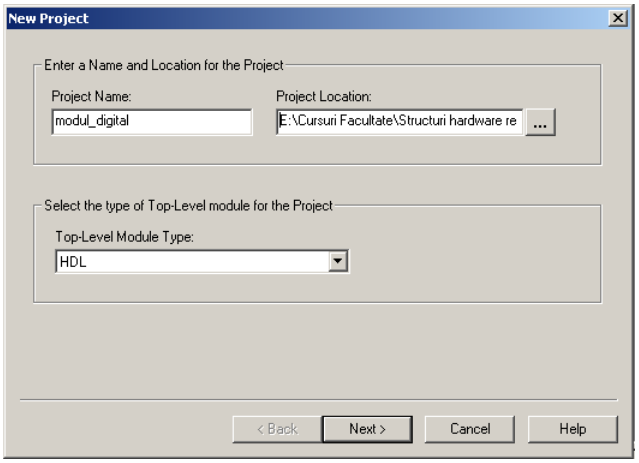
1.1. Crearea proiectului

Se lansează programul XILINX ISE din mediul Windows.

Primul pas în realizarea modulului digital constă în crearea proiectului.

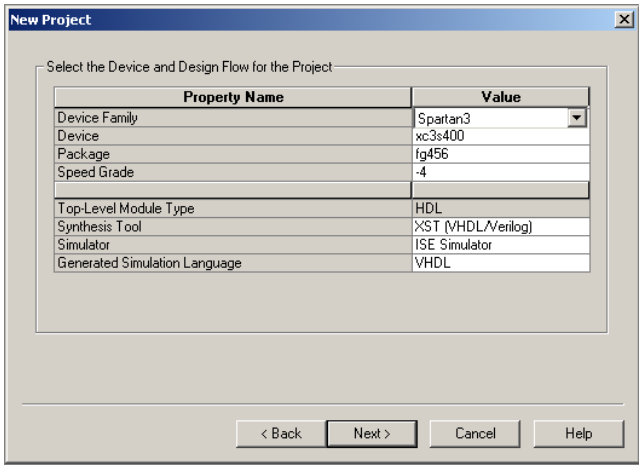
Crearea unui proiect se face plecând de la fereastra generală, prin selectarea **File -> New Project**.

Se deschide o fereastră de dialog care va fi completată după cum urmează:



Project Name – se introduce numele proiectului (modul_digital);
Project Location – se selectează directorul în care se doreşte salvarea proiectului;
Top-Level Module Type – fixat ca fiind de tipul HDL.

În acest caz, proiectul poartă denumirea „modul_digital”.
Se selectează butonul **Next,** după care apare următoarea fereastră:

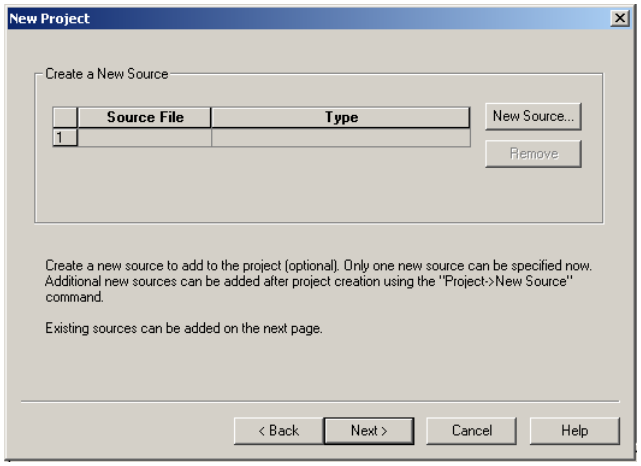


În vederea completării proprietăților ce apar în fereastră, se selectează linia corespunzătoare acestora din zona **Value** şi se completează din listele afişate următoarele valori:

- **Device Family:** structura reconfigurabilă utilizată în cadrul laboratorului este *Spartan 3*;
- **Device:** va fi introdus codul circuitului utilizat. În acest caz este *xc3s400*;
- **Package:** se indică tipul capsulei şi numărul de pini; circuitul Spartan folosit are un package de tipul *fg456*;
- **Speed Grade:** viteza de propagare a semnalului de ceas în FPGA este *-4*;
- **Syntesis Tool:** *XST[VHDL/Verilog]*;
- **Simulator**: programul ales pentru simularea şi verificarea modulului digital este *ISE Simulator*;
- **Generated Simulator Language:** *VHDL*.

Toate fişierele proiectului vor fi salvate automat într-un subdirector al acestuia. Un proiect poate avea decât un singur fişier HDL, de tip *top-level* (este practic modulul principal, capul ierarhiei, care uneşte toate celelalte submodule). Modulele digitale pot fi adăugate secvențial la proiect pentru a forma o structură ierarhică şi modulară.

Se selectează butonul **Next,** după care apare următoarea fereastră:



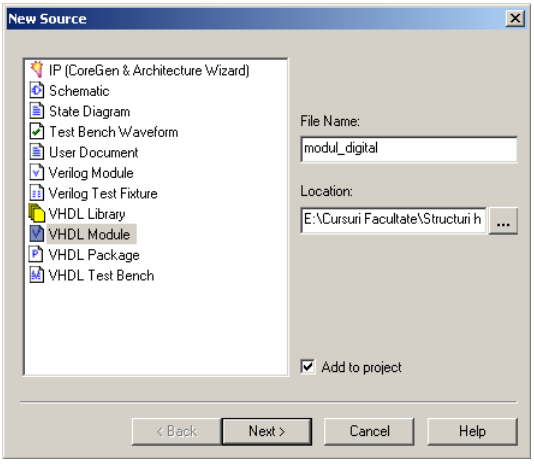
Se creează un fişier sursă nou prin selectarea butonului **New Source**.

1.2. Crearea şi introducerea unui fişier sursă VHDL

În cele ce urmează se va introduce în mediul Xilinx ISE un program în limbajul VHDL. Se creează un fişier VHDL, cu extensia **.vhd* ce va fi scris utilizând editorul programului XILINX.

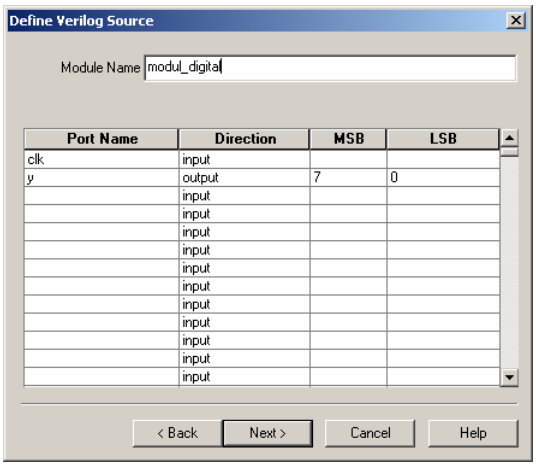
În figura de mai sus se selectează butonul **New Source** pentru a fi introdus un fişier nou. Dacă nu se doreşte acest lucru, ci doar adăugarea unor fişiere deja existente, se selectează butonul **Next**.

Va fi afişat ecranul:



După cum se poate vedea în figura de mai sus, pot fi create mai multe tipuri de fişiere ce pot fi adăugate la proiectul nostru.

În acest caz, se va selecta un fişier de tip **VHDL Module**, iar la zona de editare **File Name** se va introduce „modul_digital”. Opţiunea **Add to project** trebuie selectată permanent pentru ca fişierul creat sa fie ataşat la proiect. În continuare se selectează butonul **Next**.



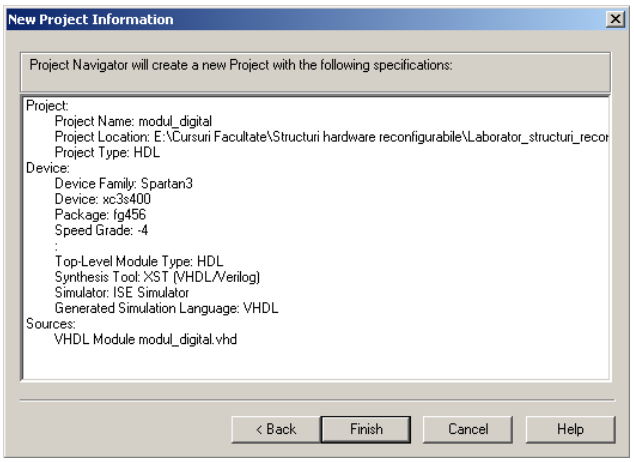
În coloana cu numele **Port Name** sunt introduse numele tuturor pinilor cu direcţia specificată în coloana **Direction** (direcţia poate fi *input*, *output*, *inout*). În cazul în care porturile sunt vectori sau magistrale, pe coloanele MSB şi LSB se specifică dimensiunea acestora.

Porturile introduse sunt următoarele:

clk	*input*	
y	*output*	*7 .. 0*

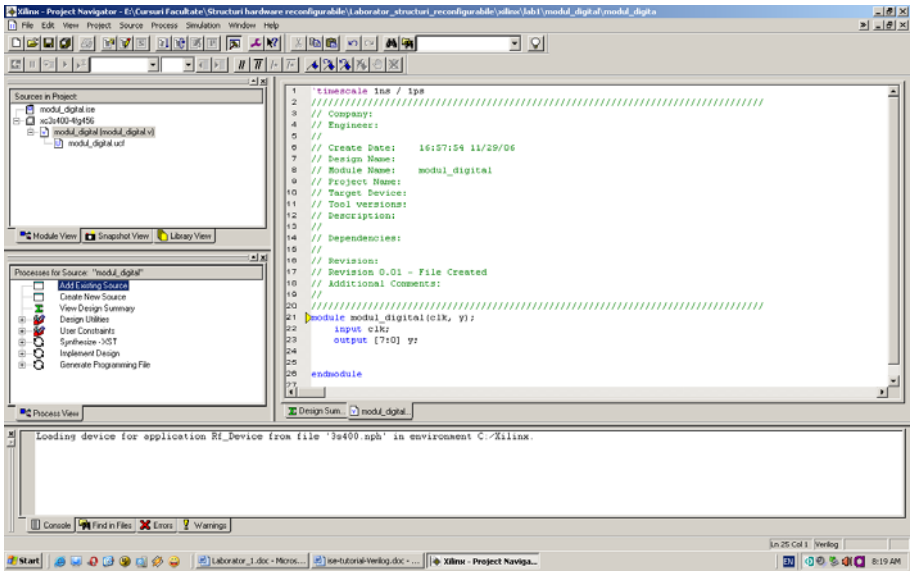
După completarea tuturor porturilor modulului digital, se selectează butonul **Next** şi apare fereastra **New Source Information,** în care sunt afişate toate datele completate până în prezent despre proiectul ce urmează a fi implementat.

Se selectează butonul **Finish**. Se revine în fereastra anterioară, dar care are adăugat fişierul deja creat. În continuare se selectează butonul **Next**. Va apare o fereastră ce dă posibilitatea utilizatorului să adauge proiecte noi. În final, după o nouă selectare a butonului **Next** va fi afişată fereastra de mai jos, în care sunt prezentate toate specificaţiile proiectului creat.



Se selectează butonul **Finish**, iar fişierul VHDL va apare în zona **Sources in Project** din **Project Navigator**.

1.3. Editarea fişierului VHDL



Fişierul sursă poate fi vizualizat în fereastra **Project Navigator**. Fereastra în care este afişat fişierul sursă poate fi utilizată ca un editor în vederea completării sau modificării programului VHDL. Este recomandat ca programul să fie salvat periodic prin comanda **File → Save** din meniul principal. Programele VHDL pot fi editate în orice editor de text şi ataşate la proiect prin comanda **Add Copy Source**.

Programul VHDL va fi completat cu liniile următoare de cod sursă:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity modul_digital is
   Port ( clk : in std_logic;
          y : out std_logic_vector(7 downto 0));
end modul_digital;

architecture Behavioral of modul_digital is
            signal temporar: std_logic_vector(2 downto 0):= (others => '0');
begin
           process(clk)
           begin
            if (clk'event and clk = '1')          then
                     temporar <= temporar + '1';
            end if;
           end process;

           process (temporar)
           begin
                     case (temporar) is
                              when "000" =>  y <= "10000001";
                              when "001" =>  y <= "01000010";
                              when "010" =>  y <= "00100100";
                              when "011" =>  y <= "00011000";
                              when "100" =>  y <= "00100100";
                              when "101" =>  y <= "01000010";
                              when "110" =>  y <= "10000001";
                              when others => y <= (others => '1');
                     end case;
           end process;
end Behavioral;
```

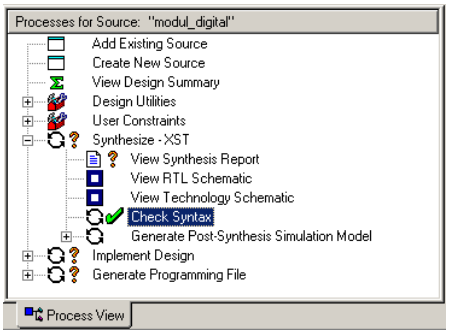
Programul este format din doua blocuri. Primul bloc are funcția de numărător pe 3 biți, iar cel de-al doilea este un decodor binar într-o formă definită de proiectant.

Legătura dintre blocuri este realizată prin doi regiştri declarați în prima parte a modulului digital (*y* şi *temporar*). Regiştrii au avantajul că pot păstra valorile salvate pe o perioadă nedeterminată de timp.

În vederea determinării corectitudinii codului sursă din fişier se va verifica sintaxa acestuia.

În cadrul ferestrei **Process View** se selectează **Check Syntax**. După verificare, dacă nu există nicio eroare va apare semnul ✔ în dreptul acesteia.

În caz contrar, erorile vor fi afişate în partea de jos a display-ului, în fereastra **Console**.



1.4. Simularea proiectului

Simularea proiectului este necesară pentru verificarea corectitudinii funcționale ale modulelor digitale create. În prima fază, simularea se face prin vizualizarea formelor de undă ale stimulilor de intrare şi ale semnalelor rezultate la ieşirile modulelor digitale. În acest scop este necesară crearea unui fişier de tip **Test Bench Waveform**, urmărind paşii de mai jos:

- Se selectează fişierul sursă *modul_digital.vhd* din fereastra **Sources in Project**;
- Se creează o sursă nouă selectând **Project → New Source**;
- În fereastra **New Source Window** se selectează fişierul de tip **Test Bench Waveform** şi i se alocă numele *fisier_test*;
- Se selectează butonul **Next** după care apare o fereastră **Select** în care vor apare fişierele sursă editate de utilizator până în prezent. În cazul nostru apare doar fişierul *modul_digital*;
- Se selectează butonul **Next** şi apoi butonul **Finish**.

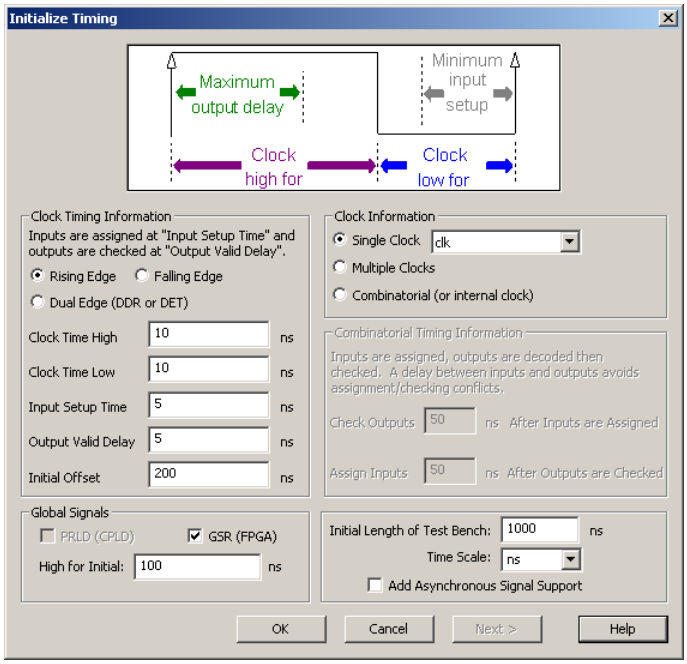
În final, apare fereastra **Initialize timing** şi se fac următoarele setări:

- în zona **Clock Timing Information** se selectează frontul de ceas pozitiv **Rising Edge**, **Clock Time High** de 10ns, **Clock Time Low** de 10ns, **Input Setup Time** de 5ns, **Output setup time** de 5ns, iar **Initial Offset** de 200ns;

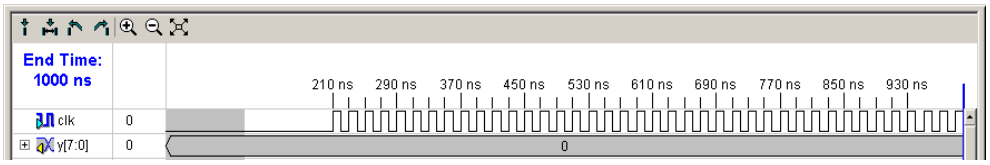
- în zona **Global Signals** se selectează **GSR(FPGA)** şi semnalul de ceas, care inițial, va fi 1 logic pentru 100ns;

- în zona **Clock Information**, se selectează un singur semnal de ceas **Single Clock** şi se introduce denumirea acestuia din programul VHDL (în cazul nostru semnalul **clk)**;

- în final, timpul de simulare **Initial Length Test Bench** va fi setat la valoarea de 1000ns.

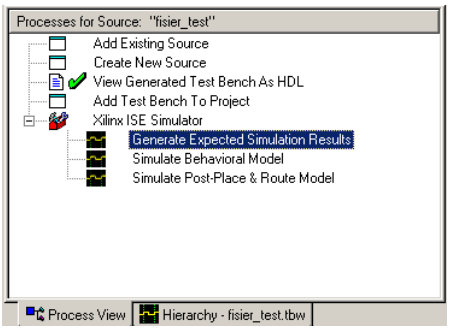


Pentru finalizarea completării acestui tabel se selectează butonul **OK**, după care apare în **Project Navigator** fereastra ce cuprinde semnele de intrare, respectiv de ieşire cu formele respective de undă.



Această fereastră va fi salvată prin comanda **File → Save** din meniu. În fereastra **Sources in Project** va apare un fişier cu numele *fisier_test.tbw*.

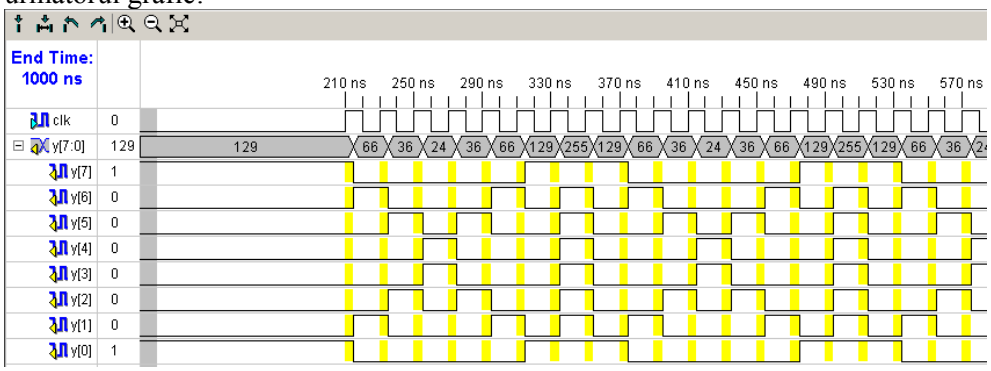
După selectarea acestui fişier, fereastra **Process View** va avea conţinutul din figura alăturată.



Prin selectarea procesului **Generate Expected Simulation Results** se realizează simularea funcţională în care nu se ţine cont de timpii de propagare prin circuit.

În acest pas vor fi afişate formele de undă ale semnalelor de ieşire, funcţie de cele ale semnalelor de intrare. Practic, fereastra de afişare prezintă funcţia pe care programul urmează să o realizeze.

După terminarea procesului de generare a rezultatelor simulării, va fi afişat următorul grafic:



Vizualizarea la nivel de bit a semnalului de ieşire se realizează prin selectarea semnului + din dreptul acestuia.

În mod similar se generează simularea comportamentală (**Simulate Behavioral Model**) şi simularea după post procesare (**Simulate Port-Place & Route Model**).

În vederea realizării modelului final al modulului digital vor fi adăugate următoarele linii de cod VHDL:

- în zona declarativă a arhitecturii se adaugă linia signal semaf :std_logic;
- se adaugă următoarele procese:

```
process(clk)
        variable cont_temp:std_logic_vector(23 downto 0);
begin
        if (clk'event and clk = '1')        then
                semaf <= '0';
                cont_temp := cont_temp+1;
                if (cont_temp = x"FAF080") then
                        cont_temp := x"000000";
                        semaf <= '1';
                end if;
        end if;
end process;

process(clk)
begin
        if (clk'event and clk = '1')        then
                if semaf = '1'    then
                        temporar <= temporar + '1';
                end if;
        end if;
end process;
```

Programul final este următorul:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity modul_digital is
   Port ( clk : in std_logic;
          y : out std_logic_vector(7 downto 0));
end modul_digital;

architecture Behavioral of modul_digital is
           signal temporar:          std_logic_vector(2 downto 0):= (others => '0');
           signal semaf :std_logic;
begin
          process(clk)
                    variable cont_temp:std_logic_vector(23 downto 0);
          begin
                    if (clk'event and clk = '1')          then
                              semaf <= '0';
                              cont_temp := cont_temp+1;
                              if (cont_temp = x"FAF080") then
                                        cont_temp := x"000000";
                                        semaf <= '1';
                              end if;
                    end if;
          end process;
          process(clk)
          begin
                    if (clk'event and clk = '1')          then
                              if semaf = '1'     then     temporar <= temporar + '1';
                              end if;
                    end if;
          end process;
          process (temporar)
          begin
                    case (temporar) is
                              when "000" => y <= "10000001";
                              when "001" => y <= "01000010";
                              when "010" => y <= "00100100";
                              when "011" => y <= "00011000";
                              when "100" => y <= "00100100";
                              when "101" => y <= "01000010";
                              when "110" => y <= "10000001";
                              when others => y <= (others => '1');
                    end case;
          end process;
end Behavioral;
```

1.5. Crearea şi editarea constrângerilor

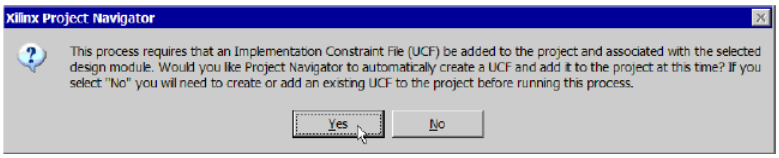
În orice proiect este necesară utilizarea constrângerilor în vederea respectării cerințelor referitoare la timpii de propagare, aria ocupată şi atribuirea pinilor. În acest scop sunt posibile trei categorii de constrângeri.

1.5.1. Constrângeri referitoare la domeniul timp

Minimal, fiecărui proiect trebuie să i se specifice constrângeri referitoare la perioada semnalului de ceas şi a timpului de offset, în vederea respectării vitezei de lucru a circuitului. Paşii de realizare a unui fişier de constrângeri sunt următorii:

- se selectează fişierul sursă *modul_digital* din fereastra **Sources in Project**; în fereastra **Process for Source** vor apare procesele corespunzătoare;
- se expandează grupul **User Constraints** în fereastra **Process for Source** şi se selectează procesul **Create Timing Constraints**.

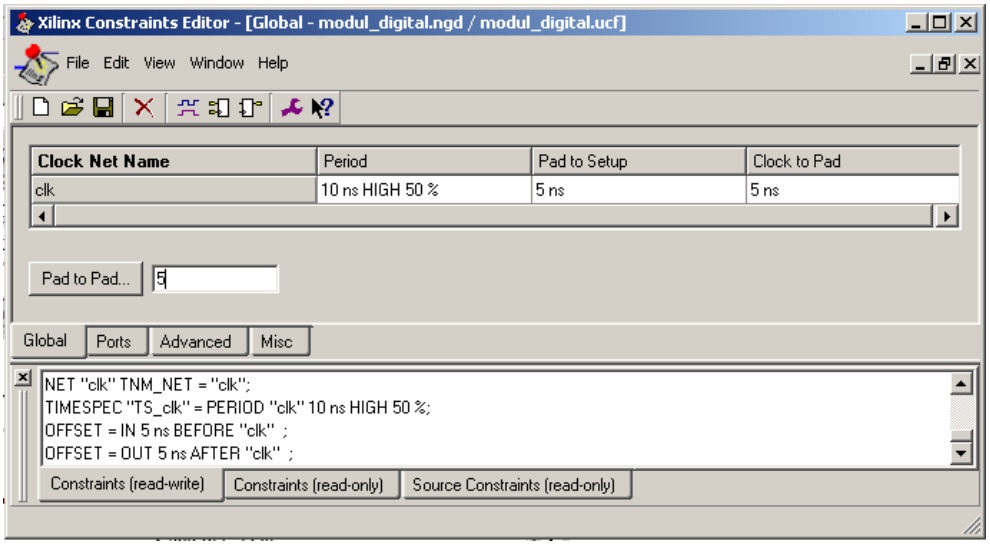
Mediul de dezvoltare Xilinx ISE rulează automat procesul de sinteză şi creează un fişier de tip UCF (User Constraints File). După apariția mesajului



se va selecta butonul **Yes,** fapt ce permite ataşarea fişierului UCF la proiect, iar în final, afişarea editorului de constrângeri.

În fereastra **Global** a editorului de constrângeri se vor introduce următoarele valori:

Period - 20ns, **Pad to Setup** - 10ns şi **Clock to Pad** - 10ns.



Se salvează fişierul prin comanda **File** → **Save** după care se închide editorul de constrângeri.

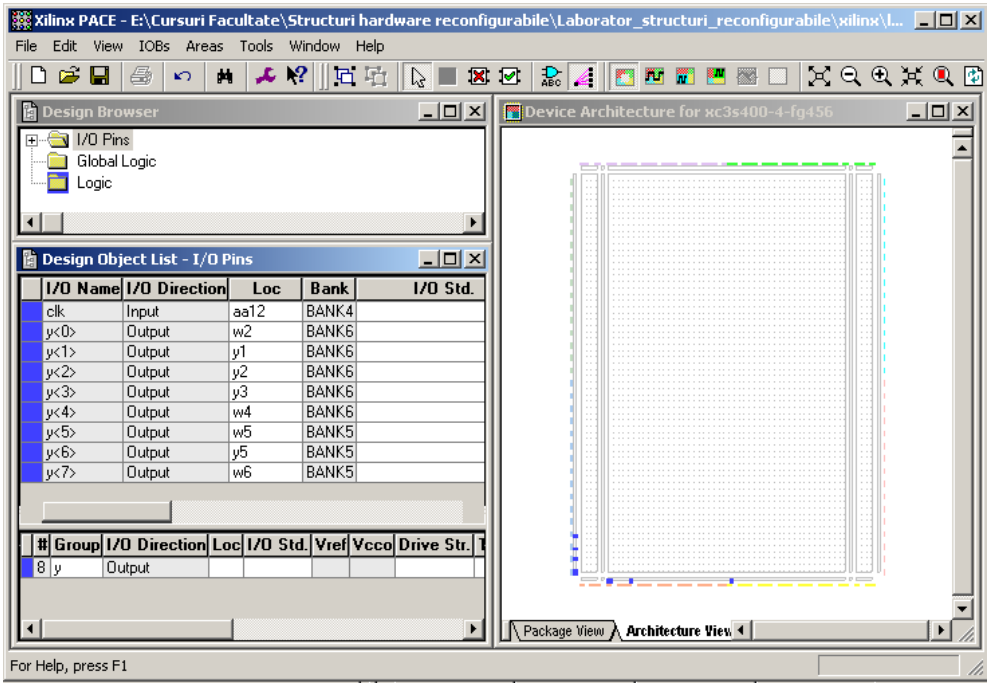
1.5.2. Constrângeri referitoare la atribuirea pinilor

Prin aceste constrângeri sunt realizate asocierile dintre porturile modulului digital declarate în entitate şi pinii circuitului de tip FPGA utilizat.

În cazul nostru vor fi introduse următoarele constrângeri:

clk → aa12
y[0] → w2
y[1] → y1
y[2] → y2
y[3] → y3
y[4] → w4
y[5] → w5
y[6] → y5
y[7] → w6

Crearea unui astfel de fişier se face prin selectarea procesului **Assign Package Pins** în grupul **User Constraints**. Programul **Xilinx Pinout and Area Constraints Editor (PACE)** se va deschide automat.



După completarea constrângerilor se selectează **File → Save**. La apariţia casetei de atenţionare se selectează **XST Default < >** şi apoi butonul **OK**. În final se închide editorul.

1.6. Sinteza proiectului

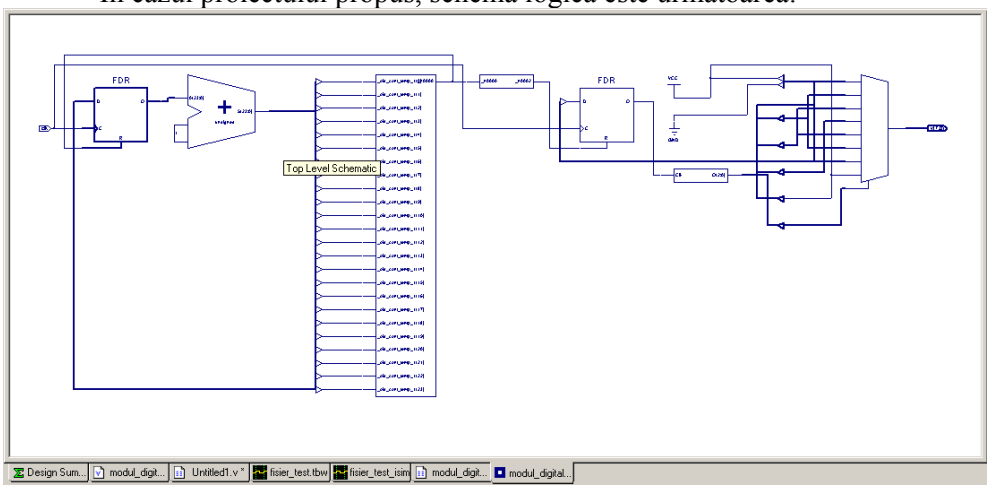
După ce proiectul a fost completat, verificat şi s-au adăugat fişierele de constrângeri, acesta poate fi sintetizat şi apoi implementat pentru a se face, în final, programarea structurii reconfigurabile.

Sinteza se realizează prin selectarea (dublu clic) a grupului **Synthesize – XST**.

Dacă procesul nu a generat erori poate fi vizualizat primul raport referitor la modulul digital. În acest scop, se selectează grupul **Synthesize – XST,** după care se expandează şi se alege **View Synthesis Report**. În cadrul raportului, printre alte informaţii, se regăsesc specificaţii referitoare la gradul de ocupare al ariei şi a timpului de propagare prin structura reconfigurabilă.

În cadrul aceluiaşi grup, dacă se selectează **View RTL Synthesis**, se vizualizează schema logică generată după sinteza modulului digital.

În cazul proiectului propus, schema logică este următoarea:



Selectarea (dublu clic) fiecărui bloc poate conduce la o descriere mai detaliată la nivel de schemă.

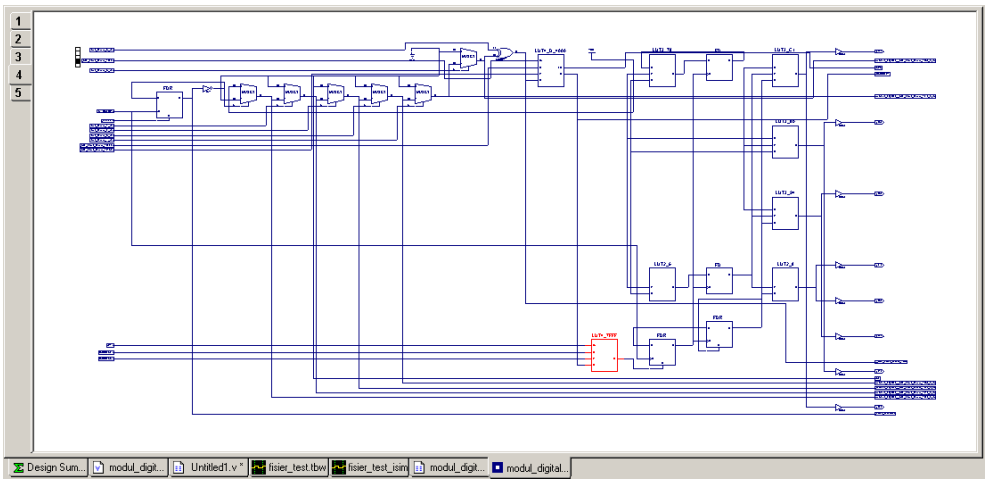
Tot în grupul **Synthesize – XST,** dacă se selectează **View Technology Schematic** se vizualizează schema tehnologică a modulului digital.

Practic structura digitală este formată din blocuri logice configurabile predefinite. Acestea, la rândul lor, sunt formate din SLICE-uri, un SLICE având în componenţă două generatoare de semnal.

În final, un generator de semnal este format dintr-un LUT (Look Up Table), memorie, multiplexoare şi bistabili. LUT-ul este elementul principal în

realizarea funcţiilor logice combinaţionale. LUT-ul pot fi comparat cu un codor ce are mai multe intrări, respectiv o ieşire la care se obţine o anumită funcţie logică.

O mică parte din schema tehnologică a modului digital este prezentată în figura de mai jos:



Dacă este selectat (dublu clic) un circuit LUT, se poate vizualiza schema logică pe care o implementează şi tabela de adevăr corespunzătoare.

1.7. Implementarea proiectului

Implementarea constă în următoarele procese: **Translate, Map** şi **Place & Route**.

Translatarea serveşte la trecerea de la schema logică rezultată în urma sintezei la primitive Xilinx.

Map-area este procesul în care circuitul este descris la nivel de componente fizice în structura FPGA:

Procesul de **Plasare şi rutare** constă în formarea fişierelor ce specifică rutările dintre componentele interne ale structurii reconfigurabile, în vederea generării fişierelor de configurare.

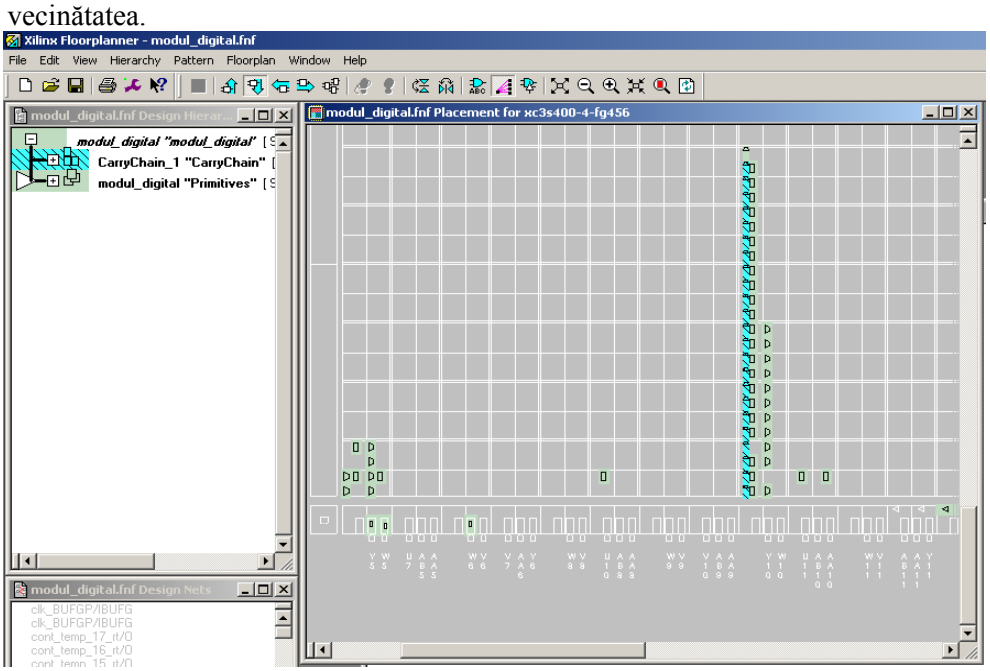
Procesul de implementare se realizează prin selectarea (dublu clic) a grupului **Implement Design**. Dacă procesul s-a terminat fără erori se poate trece la etapa de generare a fişierelor de configurare.

1.8. Verificarea procesului de implementare

Programul **Floorplanner** este utilizat pentru verificarea pinilor şi plasarea acestora. În plus, mai este folosit pentru verificarea utilizării corecte a grupurilor de celule folosite în modulul digital ce se doreşte a fi creat.

În acest scop, se selectează fişierul *modul_digital*. Se expandează grupul **Place&Route** şi se selectează (dublu clic) procesul **View/Edit Placed Design (Floorplanner)**.

Selectarea porţiunii de circuit utilizată se realizează prin comanda **View Zoom → Toolbox**. În figura de mai jos se observă primitivele utilizate în proiectul propus. Dacă se selectează una dintre acestea, apar automat interconexiunile cu vecinătatea.



1.9. Vizualizarea Resurselor Utilizate

În fereastra **Processes for Source** se selectează procesul **View Design Summary** şi deschide următoarea fereastră, în care se specifică gradul de ocupare al structurii reconfigurabile.

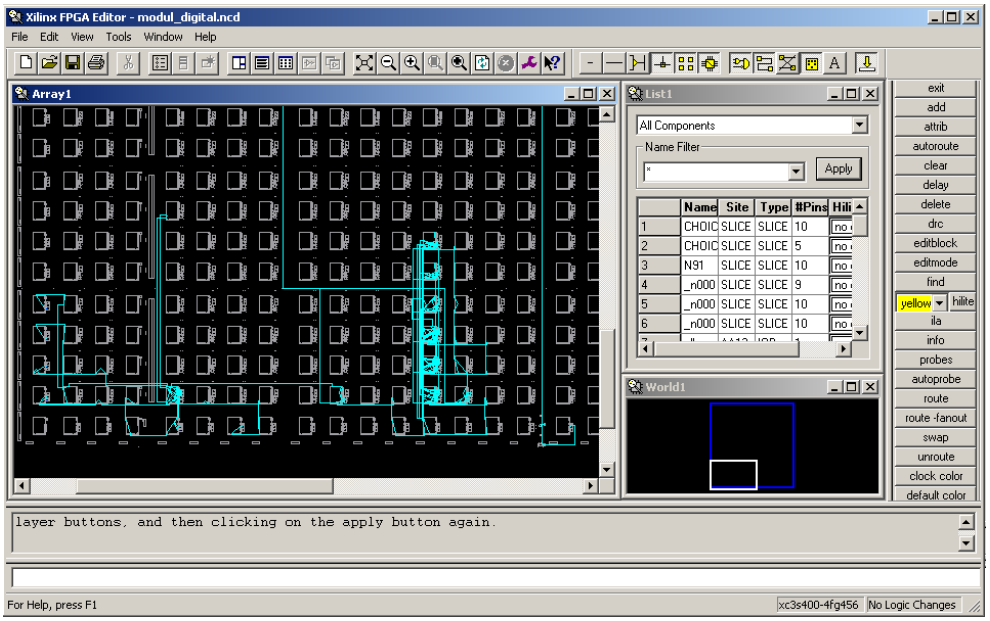
Design Overview for modul_digital

Property	Value
Project Name:	e:\cursuri facultate\structuri hardware reconfigurabile\laborator_structuri_reconfigurabile\xilinx\lab1\modul_digital
Target Device:	xc3s400
Constraints File:	modul_digital.ucf
Report Generated:	Thursday 11/30/06 at 15:25
Printable Summary (View as HTML)	modul_digital_summary.html

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops:	28	7,168	1%	
Number of 4 input LUTs:	18	7,168	1%	
Logic Distribution:				
Number of occupied Slices:	25	3,584	1%	
Number of Slices containing only related logic:	25	25	100%	
Number of Slices containing unrelated logic:	0	25	0%	
Total Number 4 input LUTs:	41	7,168	1%	
Number used as logic:	18			
Number used as a route-thru:	23			

În mod asemănător se selectează din grupul **Place & Route** procesul **View/Edit Routed Design (FPGA)**. Se selectează icoana pentru a fi evidenţiate semnalele de rutare din interiorul structurii reconfigurabile. Pentru o mai bună vizualizare se măreşte zona de interes.



Dacă se intră suficient de mult în detaliu se pot vedea rutările în interiorul CLB-urilor şi a SLICE-urilor.

1.10. Configurarea circuitului FPGA

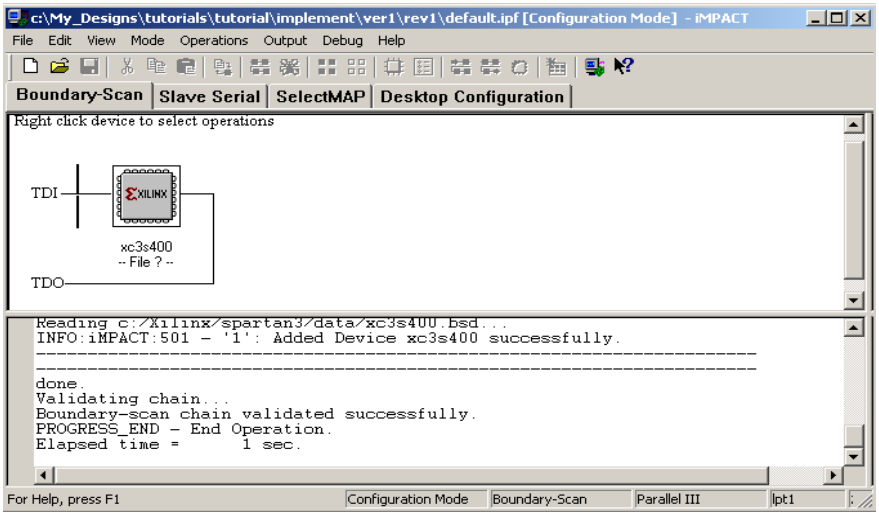
În final, pentru generarea şi implementarea fizică a fişierului binar în structura hardware reconfigurabilă, se apelează programul **IMPACT**.

În cadrul grupului **Generate Programming File** din fereastra **Processes for Source** se selectează procesul **Configure Device (iMPACT)**.

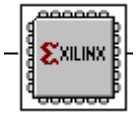
În caseta de dialog **Configure Devices** se selectează opţiunea **Boundary-Scan Mode** şi apoi butonul **Next**. Dacă apare mesajul identificării circuitului se selectează butonul **OK**.

Se continuă prin apariţia fereastrei de dialog **Assign New Configuration File**, în care se selectează fişierul binar corespunzător proiectului realizat, iar mesajul de atenţionare ce poate să apară nu va fi luat în considerare.

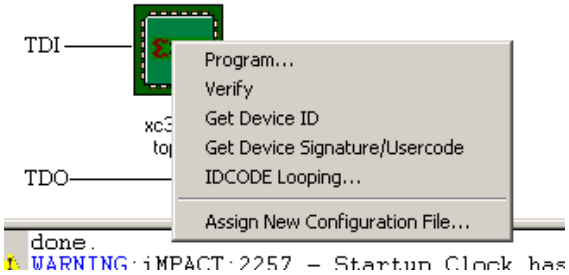
În final, apare fereastra următoare, prin care se poate realiza programarea circuitului.



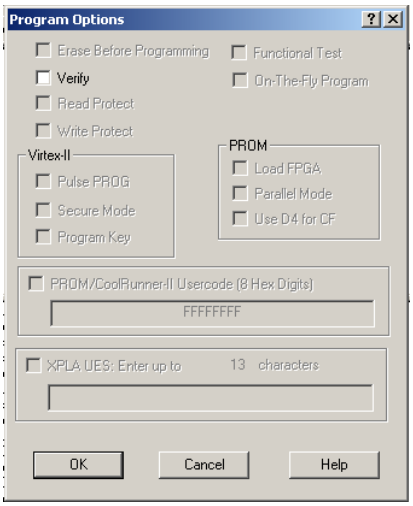
Pentru a fi introdus fişierul binar de configurare a structurii reconfigurabile se selectează pictograma:



Se ignoră mesajul de atenţionare, după care se programează circuitul FPGA prin selectarea câmpului **Program…** din figura alăturată.



Urmează apariţia fereastrei **Program Opţions**:



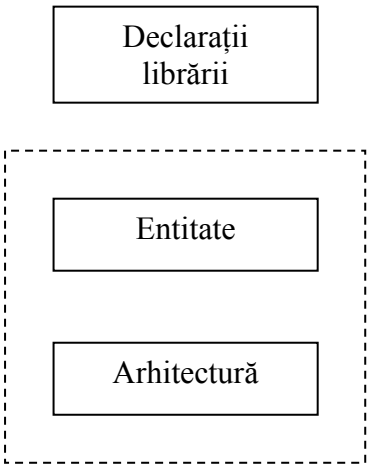
După selectarea butonului OK se porneşte procesul de descărcare a fişierului de configurare în circuit.

La terminarea descărcării, circuitul se configurează automat, iar rezultatele implementării proiectului apar la nivelul platformei de dezvoltare.

2. Structura unui program VHDL

Programul prin care se descrie un modul digital în limbajul VHDL conține trei părți:

- declararea librăriilor care vor fi utilizate în proiect;
- declararea entității modulului ce urmează a fi proiectat;
- descrierea arhitecturii acestuia.



Entitatea descrie interfața modulului digital cu semnalele din mediul exterior. O entitate deja declarată poate fi accesată de către alte entități.

Sintaxă:

```
entity nume_entitate is
        generic (listă_generică);
port (listă_de_porturi);]
        end entity nume_entitate;
```

Prin specificația **entity** se declară numele modulului digital. În plus, pot fi declarați parametrii generici şi porturi care fac parte din această entitate. Porturile declarate într-o entitate sunt vizibile în toate arhitecturile atribuite acesteia. Porturile pot fi de intrare, ieşire, buffer sau bidirecționale (**IN, OUT, BUFFER, INOUT**)

Arhitectura descrie relația dintre intrările şi ieşirile porturilor entității căreia îi este asociată. O arhitectură poate avea asociată doar o singură entitate, dar o entitate poate avea asociate mai multe arhitecturi prin configurație.

Sintaxă:

```
architecture nume_arhitectură of nume_entitate is
-- declarații în arhitectură
        begin
                --specificații_concurente
        end [ architecture ] [ nume_arhitectură ];
```

Zona declarativă a unei arhitecturi poate conține declarații de tipuri, semnale, constante, subprograme, componente şi grupuri. Specificațiile concurente

din corpul arhitecturii definesc legăturile dintre intrările şi ieşirile modulului digital pe care-l reprezintă.

Sintaxa generală a unui program în cod VHDL este următoarea:

```
--Declararea librariilor prin clauza library si use
library nume_librarie;
use nume_librarie.nume_pachet.all;

--Declararea entitatii modulului digital
entity nume_entitate is
        generic(nume generice : type := valori_initiale);
        port(nume_porturi : directie tip port);
end entity nume_entitate; --entity[93]

--Corpul arhitecturii
architecture nume_arhitectura of nume_entitate is
        declaratii arhitectura
begin
        specificatii concurente
        end architecture nume_arhitectura;
```

3. Operatori utilizați în limbajul VHDL

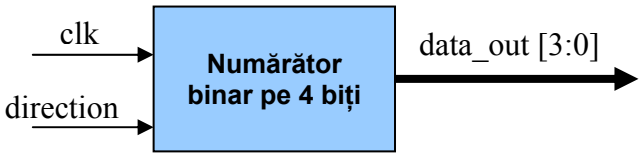
Pentru implementarea circuitelor combinaționale, în limbajul VHDL, s-au pus la dispoziția programatorului operatori logici, aritmetici, de comparație, deplasare şi de concatenare.

În tabelul de mai jos sunt prezentați, pe scurt, operatorii logici:

Tipul operatorului	Operatori	Tipul datelor
Logic	NOT, AND, NAND, OR, NOR, XOR, XNOR	BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR STD_UNLOGIC, STD_UNLOGIC_VECTOR
Aritmetic	+, -, *, /, ** (mod, rem, abs)	INTEGER, SIGNED, UNSIGNED
Comparație	=, /=, <, >, <=, >=	aproape toți
Deplasare	sll, srl, sla, sra, rol, ror	BIT_VECTOR
Concatenare	&, (, , ,)	La fel ca la operatorii logici, plus SIGNED şi UNSIGNED

III. DESFĂŞURAREA LUCRĂRII

1. Se citesc informațiile prezentate anterior şi se exemplifică pe calculatoarele existente;
2. Se va realiza şi implementa un numărător binar reversibil pe 4 biți cu porturile de intrare/ieşire prezentate în figura de mai jos:



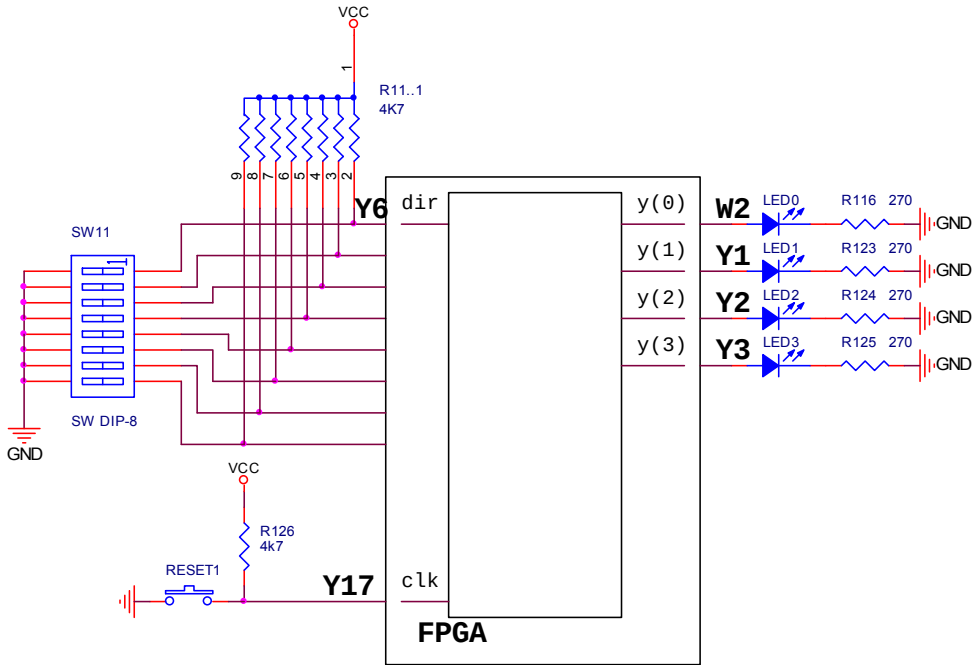
Se va utiliza următorul cod sursă VHDL al modulului digital:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter_UpDown is
   Port ( clk : in std_logic;
          direction : in std_logic;
          data_out : out std_logic_vector(3 downto 0));
end counter_UpDown;

architecture Behavioral of counter_UpDown is

begin

        process (clk)
                variable count_int: std_logic_vector(3 downto 0):= "0000";
        begin
                if (clk'event and clk = '1') then
                        if (direction = '0') then
                                count_int := count_int + '1';
                else
                                count_int := count_int - '1';
                        end if;
                end if;
                data_out <= count_int;
        end process;

end Behavioral;
```

Se vor parcurge următorii paşi:

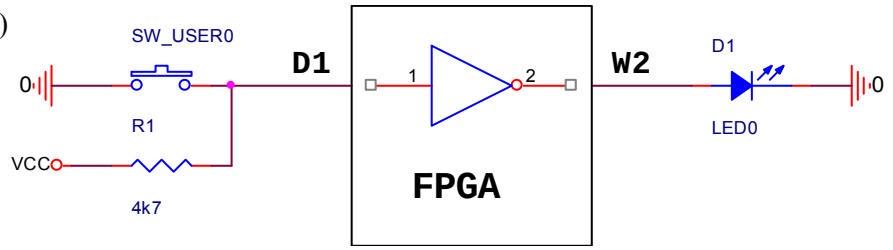
- crearea unui proiect nou;
- inserarea programului de mai sus ca sursă VHDL;
- simularea modulului digital;
- crearea fişierului de constrângeri al pinilor după schema de mai jos (fără a mai utiliza constrângeri de timp);
- realizarea sintezei acestuia;
- vizualizarea schemelor logice, respectiv tehnologice;
- simularea modelului comportamental rezultat după sinteză;
- implementarea proiectului;
- vizualizarea rulării structurii fizice;
- simularea Post-Place & Post Route a modelului;
- configurarea circuitului fizic.

Acest modul digital va fi inserat în următoarea schemă electrică:

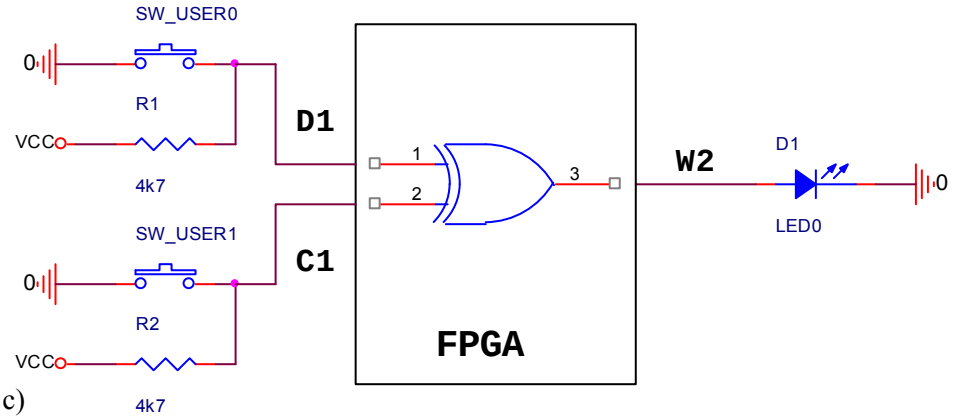


3. Respectând modelul de la punctul anterior, să se realizeze implementarea hardware a funcţiilor logice corespunzătoare pentru funcţionarea următoarelor scheme:

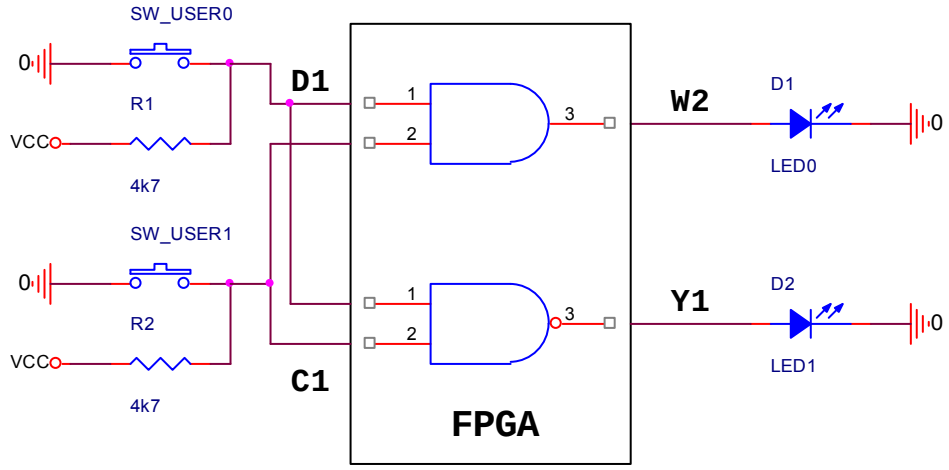
a)



b)



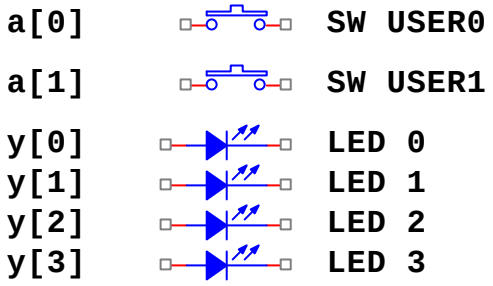
c)



4. După modelul de la punctul 2 să se implementeze hardware ecuațiile booleene ale următoarelor tabele de adevăr, în forma directă şi forma canonică.

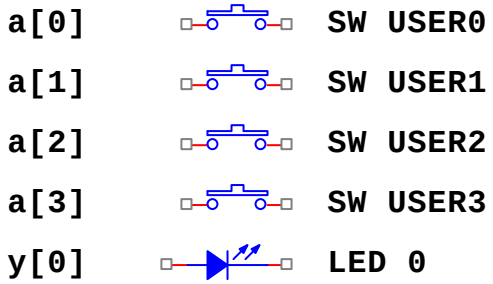
a) Tabela de adevăr pentru funcția care realizează codarea unui numărător binar pe 2 biți:

a[0]	a[1]	y[0]	y[1]	y[2]	y[3]
0	0	0	0	0	1
0	1	0	0	1	1
1	0	0	1	1	1
1	1	1	1	1	1



b). Tabela de adevăr pentru funcția care verifică paritatea impară a unui număr binar pe 4 biți:

a[0]	a[1]	a[2]	a[3]	y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



Se va realiza corespondența dintre porturile entităților şi pinilor circuitului programabil, după configurațiile din partea dreaptă a tabelelor de adevăr de mai sus.