Sisteme cu F. P. G. A. și D. S. P.

– Metode și soluțiilor actuale de progamare prin limbaje hardware –

I. INTRODUCERE:

După cum a fost prezentat în cadrul documentației anterioare, arhitectura unui sistem de calcul pe bază de FPGA este **re-configurabilă**, și conține diverse elemente logice ne-grupate între ele. Conexiunile dintre aceste elemente se realizează prin intermediul unor matrici de tranzistoare sau arii de memorii (sRAM, FLASH, EEPROM, eFuse etc.)

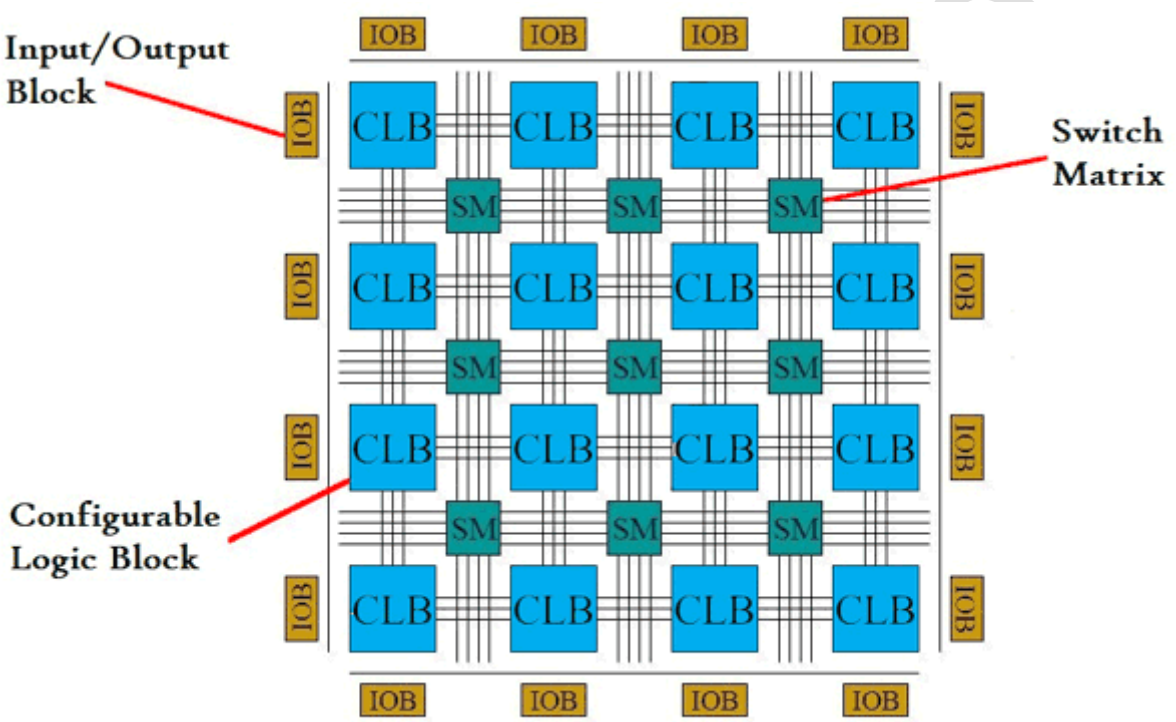


Fig. 1 – Arhitectura internă simplificată a unui nucleu FPGA [3]

De asemenea (în cadrul documentației anterioare) se amintea și faptul că, pentru a implementa aplicații la nivel de FPGA, există mai multe metode de programare precum:
- devzoltarea programului în limbaj VHDL / Verilog;
- dezvoltarea programului sub forma unui circuit cu porți logice;
- dezvoltarea programului sub forma unui model matematic Matlab – Simulink;

Trebuie menționat de asemenea faptul că programarea și re-configurarea propriu – zisă a unui sistem de calcul pe bază de FPGA decurge în mai multe etape precum:
- conceperea codului – program în limbaj VHDL / Verilog;
- sintetizarea codului la nivel de FPGA și structuri logice fizice (planificarea conexiunilor);
- implementarea fizică (planificarea stărilor comutatoarelor din matricile de conexiuni);
- generarea fișierului executabil „bit” (conține instrucțiunile de re-configurare);
- încărcarea în memorie și executarea fișierului „bit”;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Platformele de dezvoltare produse de compania Digilent, au în componența lor în mod preponderent nuclee FPGA de tip Xilinx. Xilinx reprezintă unul dintre producătorii principali de soluții pe bază de FPGA inclusiv, platforme de dezvoltare pentru domeniul academic și de cercetare sau industrie. Compania Xilinx, a dezvoltat pe lângă numeroasele variante constructive de nuclee FPGA și un mediu dedicat de programare, sintetizare și implementare fizică (eng. hardware) numit Vivado Design Suite. Predecesorul mediu a fost numit ISE Design Suite și a funcționat până aproximativ în jurul anilor 2013 – 2014, când au apărut noile generații de platforme FPGA compatibile în mod preponderent cu Vivado Design Suite.

II. PROGRAMAREA PLATFORMEI DE DEZVOLTARE BASYS 3 – FPGA XILINX ARTIX – 7:

În cadrul acestui material, vor fi prezentate principalele metode de programare ale unui sistem de calcul pe bază de FPGA de tip BASYS 3 – Artix – 7.

Fig. 2 - Platforma de dezvoltare Digilent Basys 3 – FPGA Xilinx Artix – 7[7]

Astfel se vor avea în vedere următoarele metode de programare:
- Utilizarea mediului Vivado Design Suite și a limbajului hardware Verilog;
- Utilizarea mediului Vivado Design Suite și a limbajului hardware VHDL;
- Utilizarea mediului Matlab – Simulink împreună cu System Generator (generator de cod);

Prin urmare, se vor aborda în mod comparativ toate cele trei metode de programare și configurare pentru o aplicație ce simulează funcționarea unei porți logice de tip „ȘI” (AND).
IMPORTANT! Este necesar „vivado-boards-master.zip”: https://github.com/Digilent/vivado-boards/archive/master.zip?_ga=2.83082454.153717331.1590256507-1430649020.1590256507

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

1. Implementarea unui circuit logic pe baza porții „ȘI” prin intermediul limbajului Verilog:

Pentru început se va lansa în execuție mediul Xilinx Vivado Design Suite:

Fig. 3 – Prima pagină a mediului Vivado Design Suite

- Se va alege opțiunea „Create New Project”:

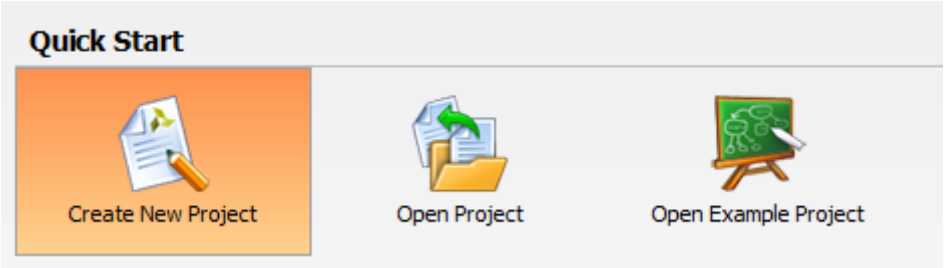


Fig. 4 – Opțiunea „Create New Project”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Se va alege opțiunea „Next” în cadrul ferestrei de inițializare a proiectului:

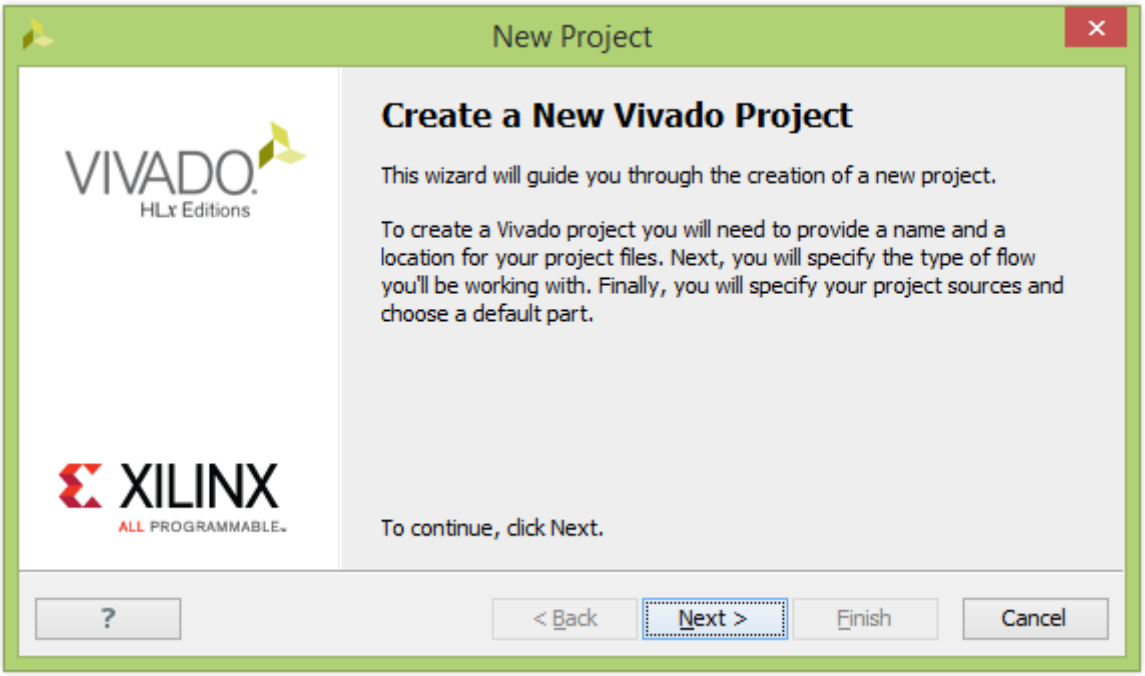


Fig. 5 – Fereastra de inițializare a proiectului

- Se va specifica numele proiectului ca și „verilog_and”, iar calea implicită de acces se va păstra „C:/Vivado_projects”. Se va bifa căsuța „Create project subdirectory”. Se va alege „Next”:

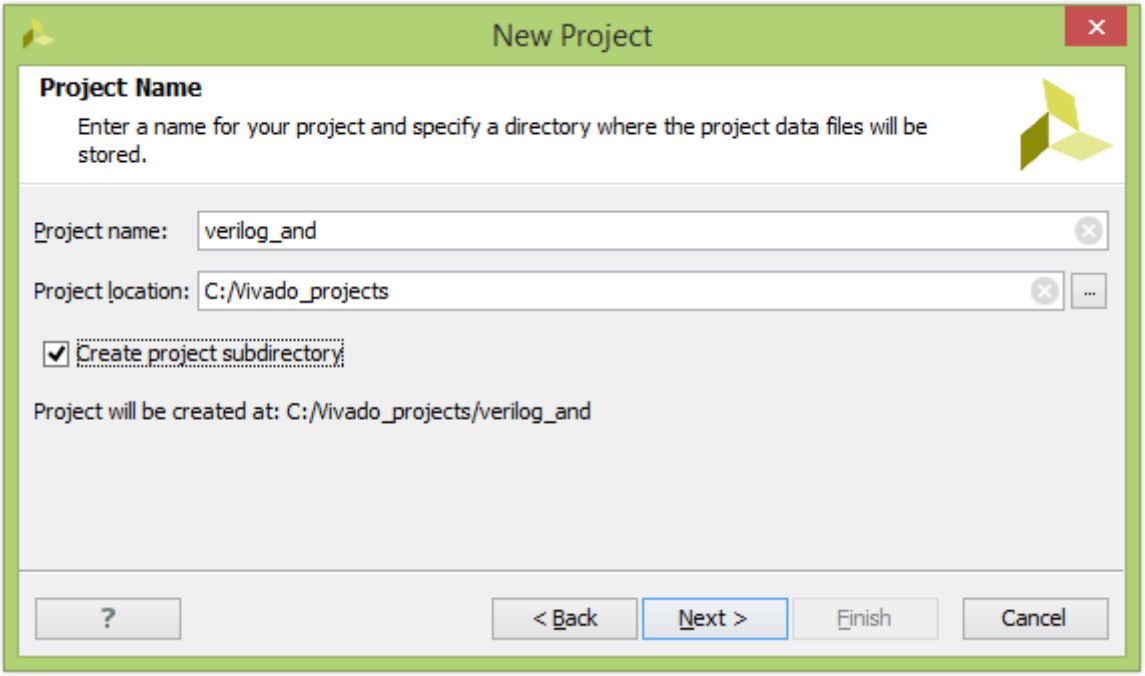


Fig. 6 – Declararea directorului spațiului de lucru

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Se va alege opțiunea „RTL Project” ca și tip de proiect „RTL”, apoi „Next”

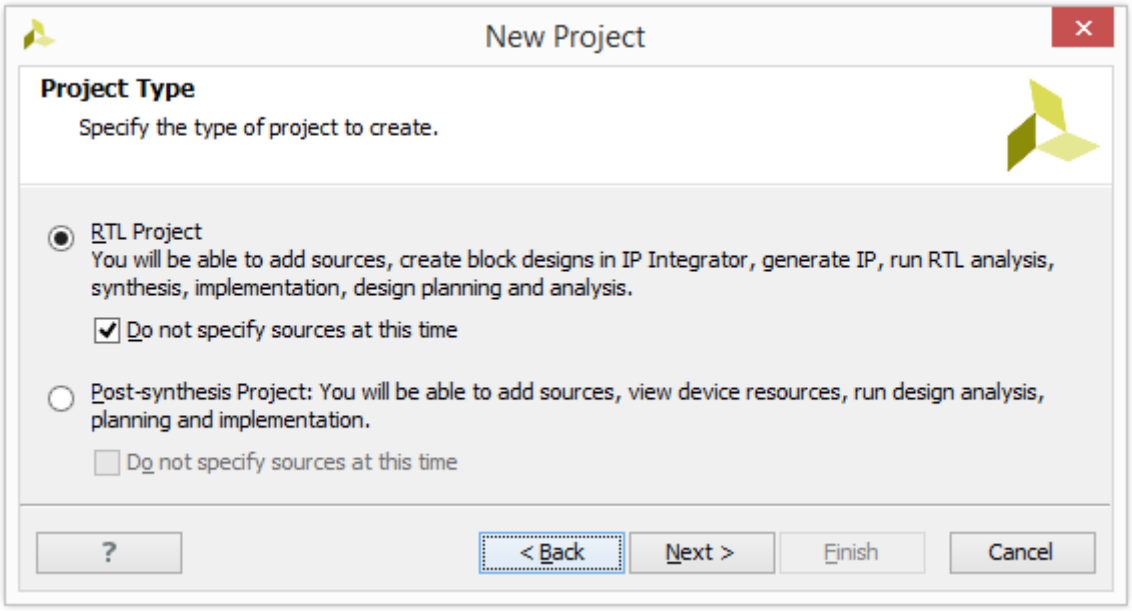


Fig. 7 – Specificarea tipului de proiect

- Se va alege categoria „Boards” iar prin intermediul casetei de căutare se va alege „Basys3”:

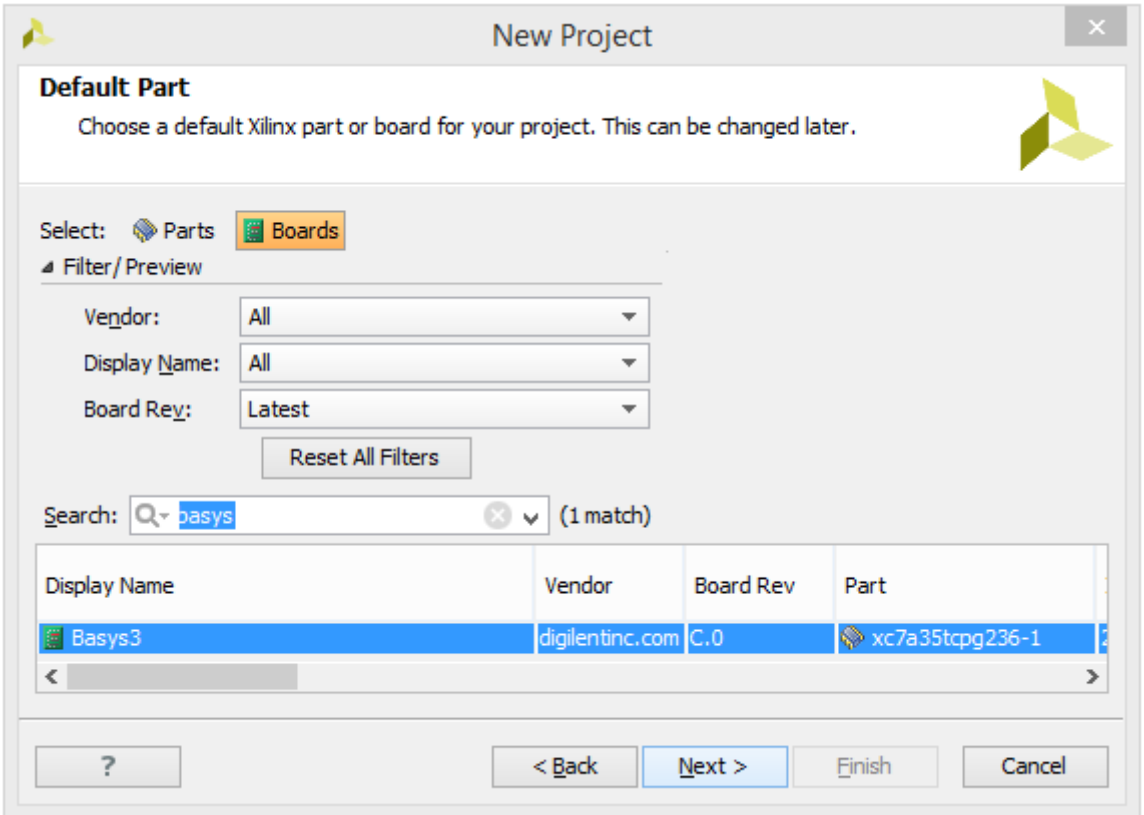


Fig. 8 – Selectarea platformei de dezvoltare BASYS 3

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

IMPORTANT! Pentru a putea găsi platforma de dezvoltare „Basys3” în listă, va trebui ca inițial pachetul de suport „vivado-boards-master.zip” să fie dezarhivat în directorul „C:\Xilinx\Vivado\2016.2\data\boards” conform instrucțiunilor oficiale Digilent: https://reference.digilentinc.com/reference/software/vivado/board-files?redirect=1

- În cadrul pasului final se va alege opțiunea „Finish” pentru a finaliza procesul de inițializare:

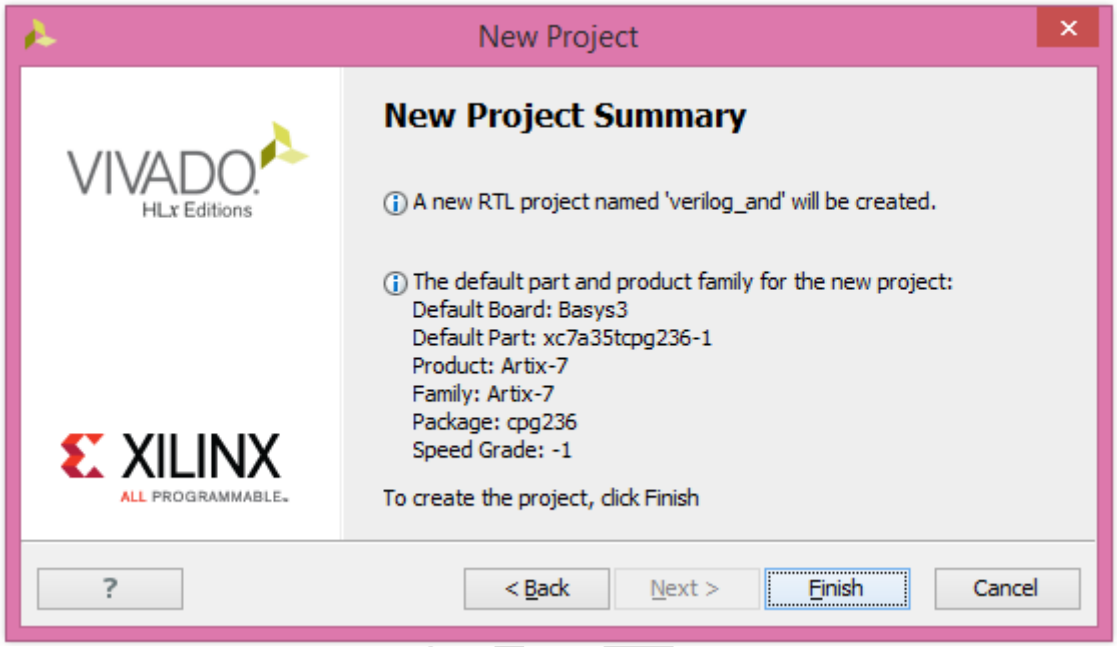


Fig. 9 – Finalizarea procedurii de adăugare a platformei Basys 3 în proiect

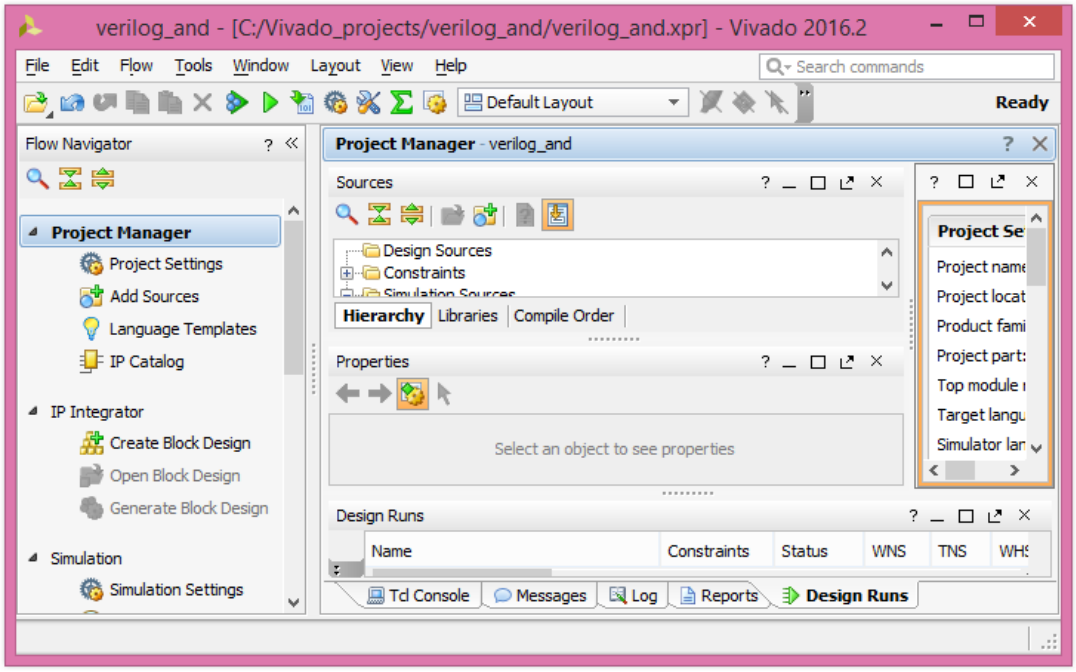


Fig. 10 – Spațiul de lucru al mediului Vivado Design Suite

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În urma parcurgerii procesului de inițializare a proiectului, spațiul grafic de lucru al mediului Vivado, se va configura în așa fel încât să poată servi în mod sugestiv și intuitiv cerințele utilizatorului pentru tipul de proiect ales (similar mediului TI Code Composer Studio).

În cadrul etapelor următoare, se vor adăuga fișierele necesare proiectului inițiat, procedeu specific mediilor de dezvoltare și programare bazate pe acest tip de abordare al progeamului (modul de lucru pe bază de proiect). Astfel, în cadrul oricărui astfel de mediu de programare, există o fereastră sau zonă în care se gestionează fișierele componente. În cadrul mediului Vivado, zonă de gestiune a fișierelor este denumită „Sources”.

Pentru a scrie codul – program în limbaj „Verilog” se va adăuga un fișier sursă de tip „Verilog”. Cu ajutorul comenzii „clic dreapta” asupra opțiunii „Design Sources” se va deschide un meniu din care se va alege opțiunea „Add Sources...”:

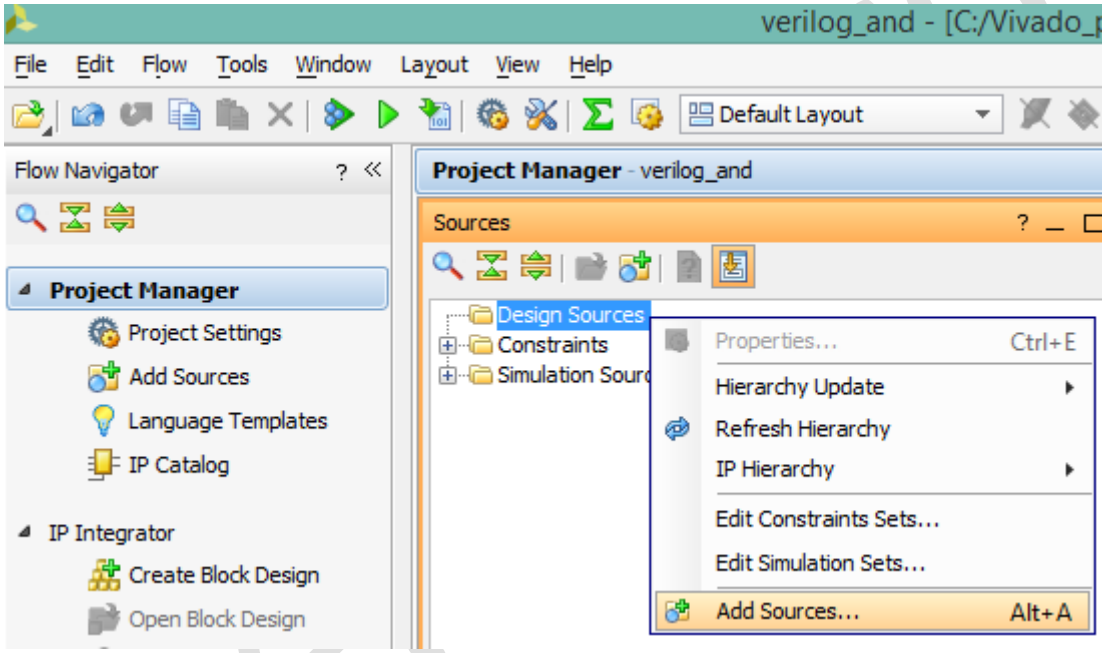


Fig. 11 – Adăugarea unui fișier sursă în cadru proiectului

Fig. 12 – Definirea procedurii de introducere a fișierului sursă

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Se va alege procedura utilizată pentru introducere a fișierului sursă. În cazul de față opțiunea „Add or create design sources” este cea mai adecvată pentru cerințele proiectului.

- În cadrul următoarei etape, se va creea fișierul sursă cu ajutorul opțiunii „Create File”:

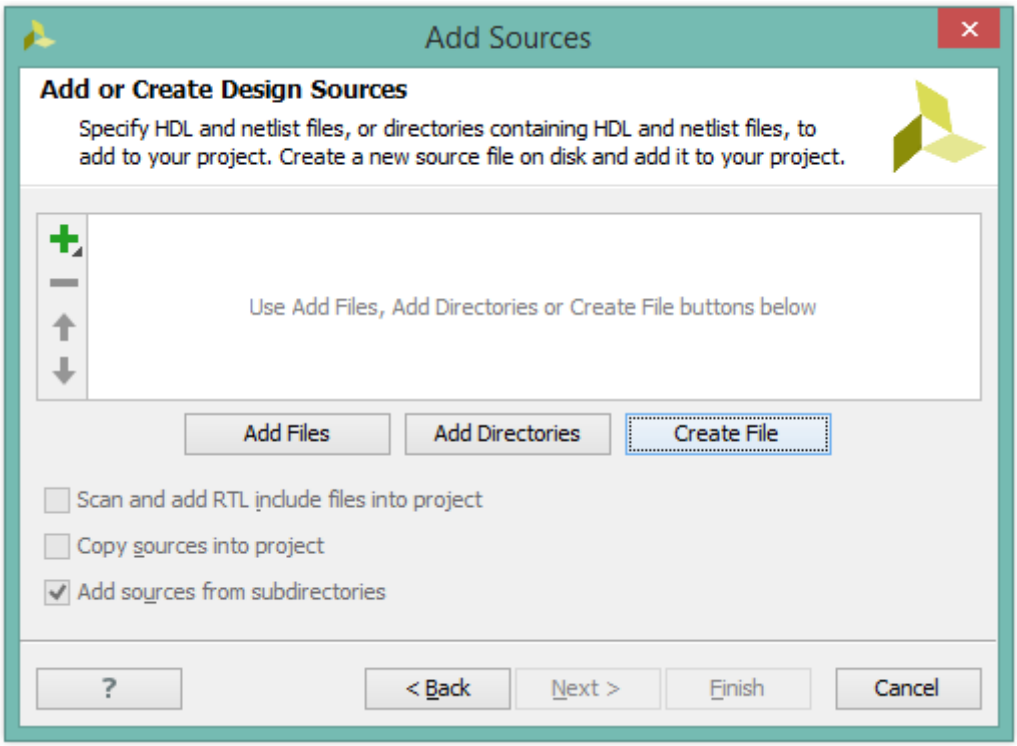


Fig. 13 – Crearea fișierului sursă

- În etapa următoarea se va specifica tipul fișierului sursă ca și „Verilog”, numele fișierului sursă ca și „Verilog_AND_gate”, iar locația sau apartenența la proiect se va alege „Local to Project”:

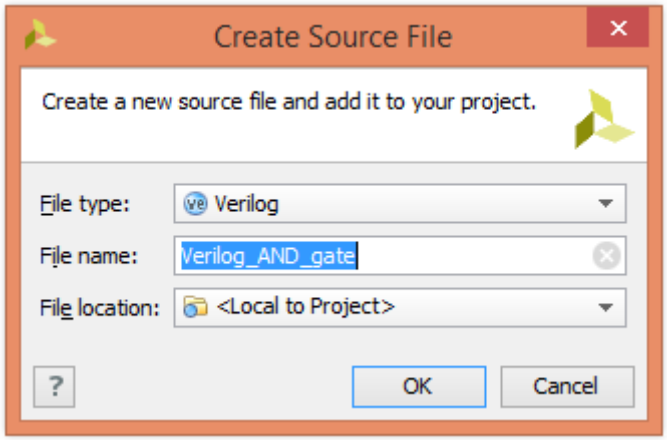


Fig. 14 – Definirea tipului de fișier sursă, denumirea și stabilirea apartenenței la proiect

- Se va finaliza etapa de definire a fișierului sursă prin apăsarea butonului „Ok” apoi „Finish”;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- După finlizarea etapei de definire a fișierului sursă Verilog, se va deschide fereastra pentru definire a porturilor fizice de intrare și ieșire utilizate în cadrul fișierului sursă (adică în codul program). Aceste porturi sunt similare variabilelor utilizate pentru denumirea intrărilor și ieșirilor digitale din cadrul microcontrolerelor sau sistemelor DSP. În cadrul acestei etape se vor alege trei porturi, anume două intrări „A” și „B” și o ieșire „f_AB”. Se va confirma introducerea parametrilor prin apăsarea butonului „Ok”:

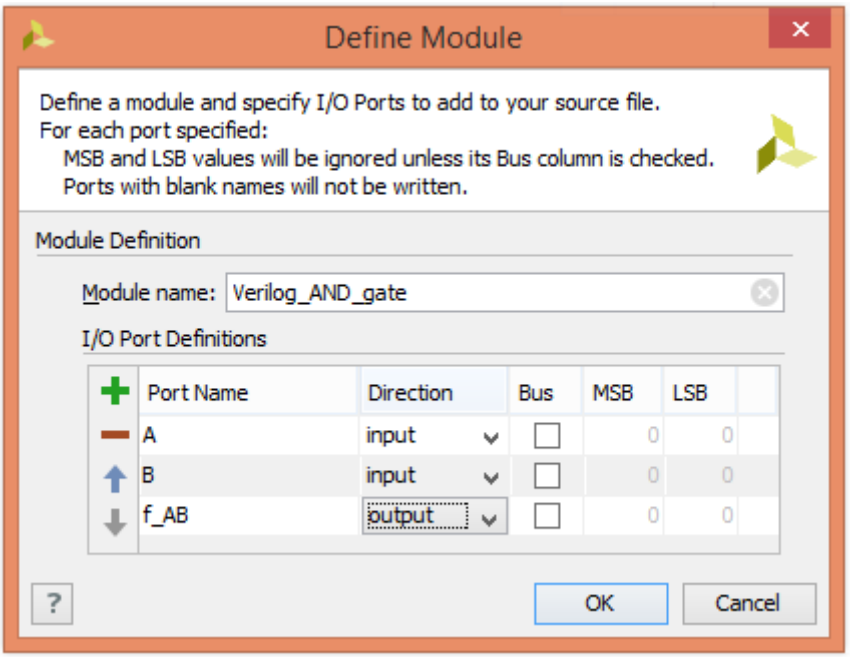


Fig. 15 – Definirea porturilor utilizate în codul – program

În cadrul ferestrei pentru gestiune a fișierelor se va regăsi și fișierul sursă scris în limbaj Verilog. Prin intermediul comenzii „dublu clic” asupra fișierului Verilog, se va putea apela editorul de cod Verilog în care se vor putea declara alte instrucțiuni suplimentare.

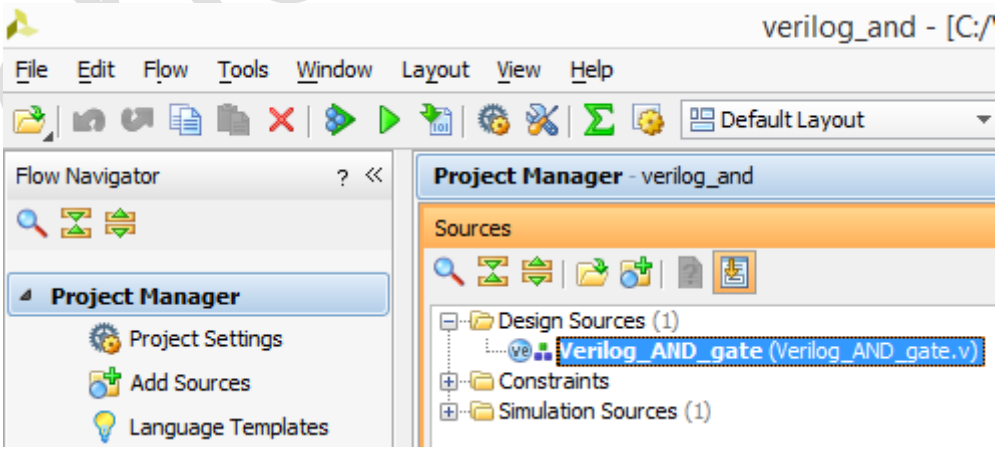


Fig. 16 – Fișierul sursă Verilog în cadrul listei componentelor proiectului

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pe lângă fișierele sursă, în cadrul proiectului, sunt necesare și fișierele pentru definire a constrângerilor „Constraints”. Constrângerile reprezintă terminalele fizice ale capsulei FPGA alocate elementelor sau resurselor externe în cadrul platformei de dezvoltare (inidcator luminoși cu LED, butoane, comutatoare, afișaj cu 7 segmente, convertor analog – digital etc.). Fișierul pentru definirea constrângerilor se poate obține de la adresa: https://www.xilinx.com/support/documentation/university/Vivado-Teaching/HDL-Design/2015x/Basys3/Supporting%20Material/Basys3_Master.xdc.
Fișierul „Basys3_Master.xdc” se va descărca prin intermediul comenzii „Save link as” și se va salva în directorul „C:\Xilinx\Vivado\2016.2\data\boards\board_files\basys3\C.0”.

În cadrul proiectului nu va fi nevoie de întreg conținutul fișierului „Basys3_Master.xdc”. Astfel, pe baza fișierului original „Basys3_Master.xdc”, se va construi un fișier particular pentru constrângerile specifice proiectului. Fiind vorba despre o singură poartă logică, în cazul de față, se vor utiliza două două intrări și o singură ieșire, anume, comutatoarele „V17” și „V16” și indicatorul luminos LED „U16”. Din cadrul fișierului general pentru constrângeri „Basys3_Master.xdc” se vor extrage doar informațiile necesare în cadrul proiectului actual. Pentru a defini un nou fișier pentru constrângeri, cu ajutorul comenzii „clic dreapta” asupra categoriei „Constraints” din cadrul listei de fișiere se va alege opțiunea „Edit Constraints Sets”:

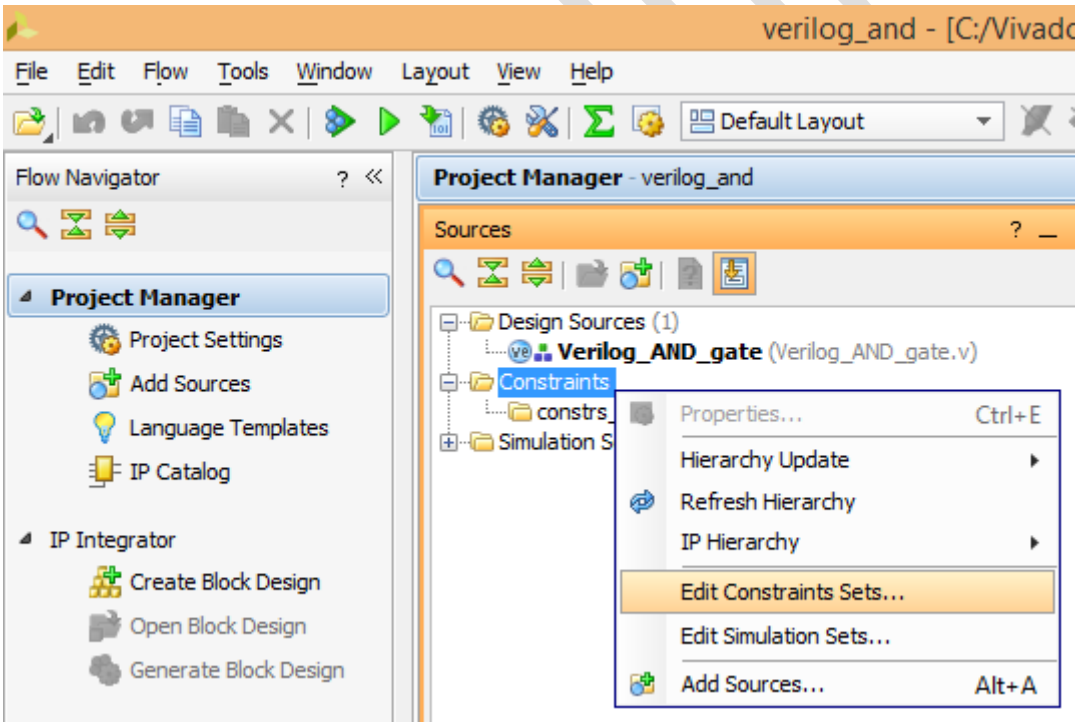


Fig. 17 – Crearea unui nou fișier pentru definirea constrângerilor

- În cadrul ferestrei nou deschise „Edit Constraints Sets” se vor putea defini fișierele particulare pentru constrângeri. Cu ajutorul opțiunii „Create File”, se va crea un nou fișier pentru constrângeri. În fereastra nou deschisă se va alege tipul fișierului (eng. File type) „XDC”, numele fișierului (eng. File name) „verilog_and_constraints”, iar locația sau apartenența fișierului (eng. File location) se va alege „Local to Project”.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

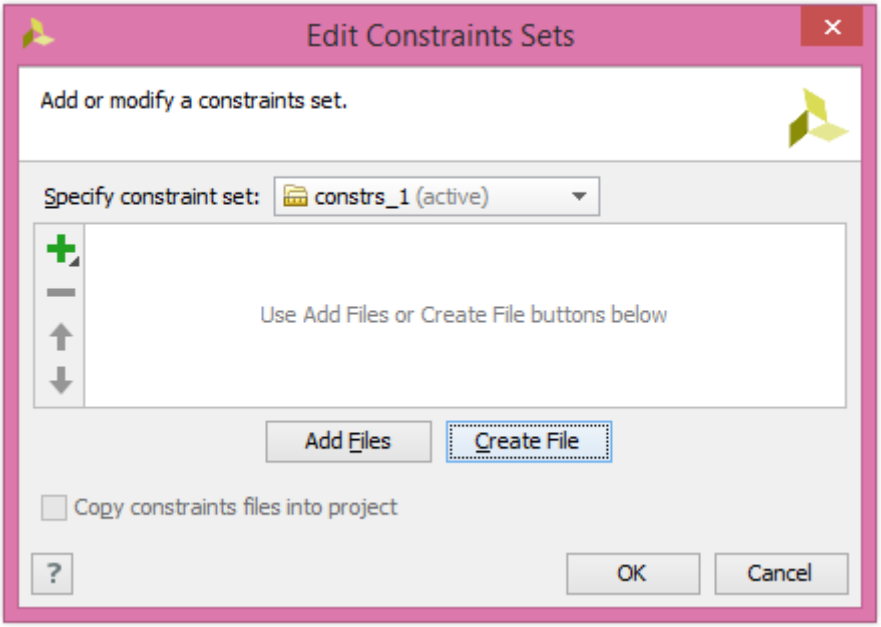


Fig. 18 – Adăugarea unui fișier pentru definirea constrângerilor în cadrul proiectului

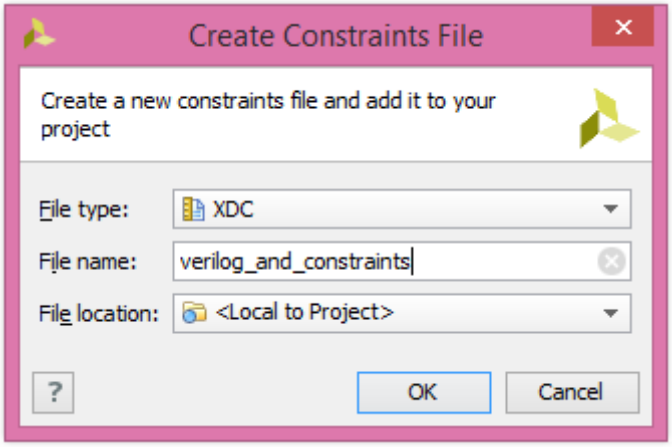


Fig. 19 – Denumirea fișierului de constrângere

- Finalizarea procedeului se va realiza în cazul ambelor ferestre prin opțiunea „Ok”. În lista fișierelor, va apărea noul fișier. Comanda „dublu clic” asupra numelui va iniția editorul.

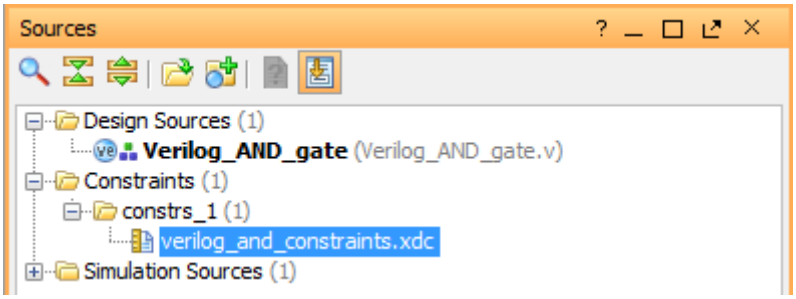


Fig. 20 – Fișierul nou pentru definirea constrângerilor la nivelul proiectului

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- În partea dreaptă a spațiului de lucru, se va deschide fereastra editorului de fișiere.

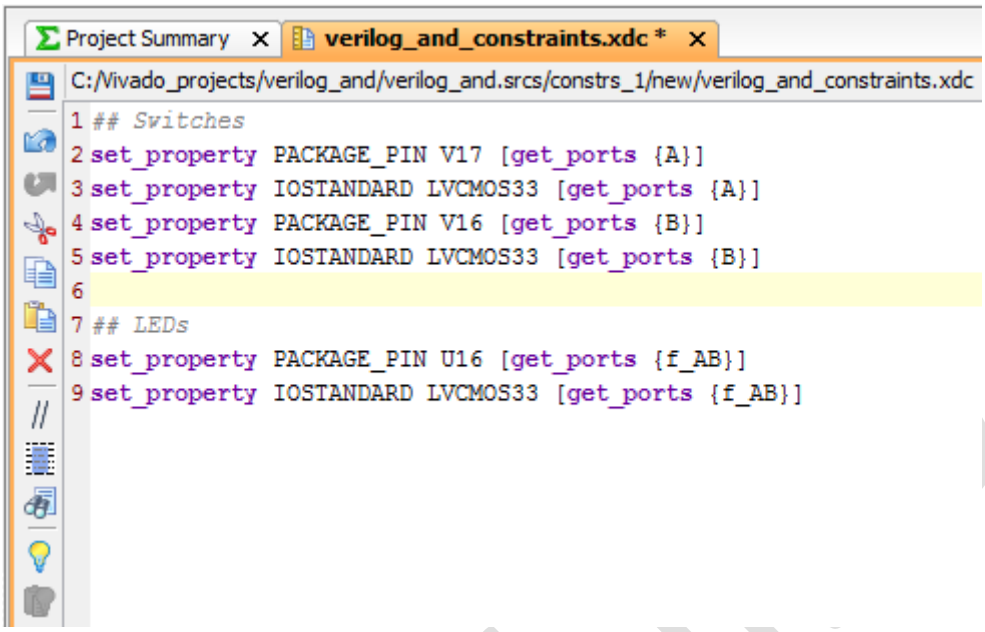


Fig. 21 – Conținutul fișierului „verilog_and_constraints.xdc”

- În cadrul acestei ferestre – editor se va specifica conținutul fișierului pentru constrângeri. Intrucțiunile din cadrul conținutului fișierului de constrângeri se va prelua din cadrul „Basys3_Master.xdc”. În cadrul instrucțiunilor „get_ports” se vor specifica (între acolade) numele porturilor declarate la inițierea fișierului Verilog sursă. Astfel, portul „A” va fi atribuit primei intrări a porții logice „ȘI” (AND), anume comutatorului fizic „V17”, portul „B” va fi atribuit întrării celei de-a doua, anume comutatorul fizic „V16”, iar ieșirea porții „f_AB” va fi atribuită unui indicator luminos LED, anume indicatorul „U16”.

În continuare, în mod similar, se vor parcurge etapele de editare a fișierului Verilog:

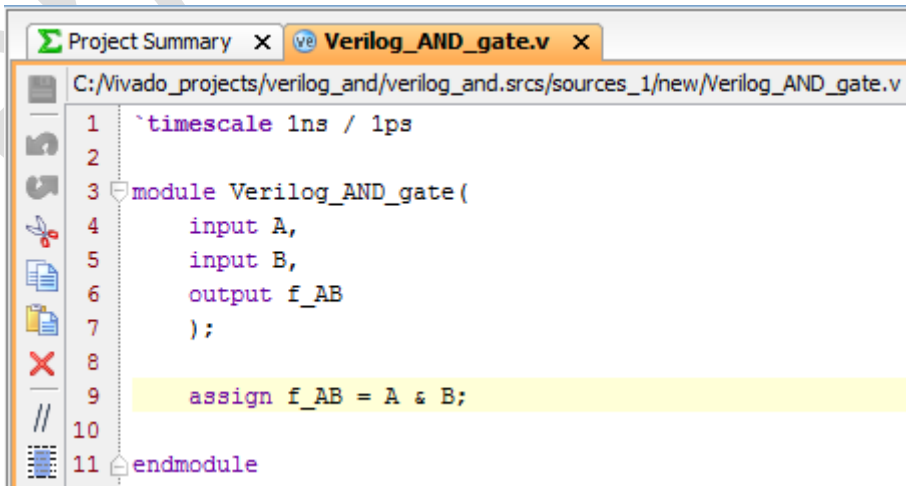
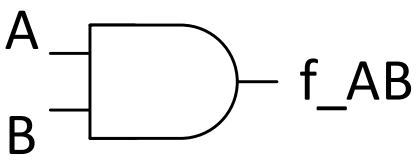


Fig. 22 – Structura fișierului sursă Verilog – codul – program

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Prin intermediul funcției „`timescale” se poate stabili baza de timp a „circuitului digital” implementat prin intermediul limbajului Verilog. Funcția „module” definește o entitate logică reconfigurabilă. În cazul de față, entitatea logică reprezintă o poartă „ȘI” (AND), care are două intrări „A” și „B” și o ieșire „f_AB”. Funcția logică specifică entității, este implementată prin intermediul instrucțiunii „assign f_AB = A & B”.



A	B	f_AB
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 23 – Funcția specifică entității logice (poartă logică ȘI)

- Comportamentul entității logice implementate este cel al unei porți logice „ȘI”. Adică, indicatorul luminos LED „U16” va fi aprins, dacă și numai dacă ambele intrări „A” și „B” sunt în stare activă (HIGH). Astfel, pentru a aprine indicatorul luminos, va trebui ca ambele comutatoare să fie în poziția activă (adică pornite).

În urma implementării codului Verilog și a stabilirii constrângerilor la nivel de entitate logică elementară, se va proceda la pasul următor, adică cel de **sintetizare** al codului Verilog. Procesul de sistentizare transpune codul Verilog într-un circuit cu porți logice implementabil la nivel de „FPGA” cu ajutorul elementelor logice disponibile în interiorul capsulei FPGA. Procesul de sintetizare este similar cu procesul de compilare, și poate fi inițiat din panoul de gestiunare al proiectului (eng. Project Manager), anume opțiunea „Run Synthesis”:

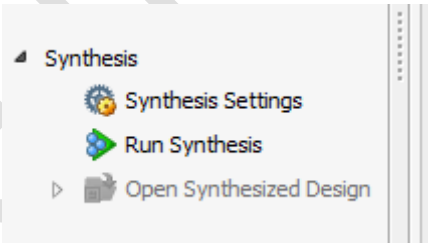


Fig. 24 – Inițierea procesului de sintetizare

- Procesul de sintetizare poate dura mai mult timp, dar la finalul acestuia se va obține modelul de configurare al elementelor logice din cadrul capsulei FPGA. Pentru a vizualiza rezultatul procesului, se va alege opțiunea „Open Synthesized Design” din cadrul ferestrei nou deschisă.

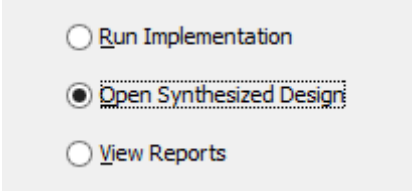
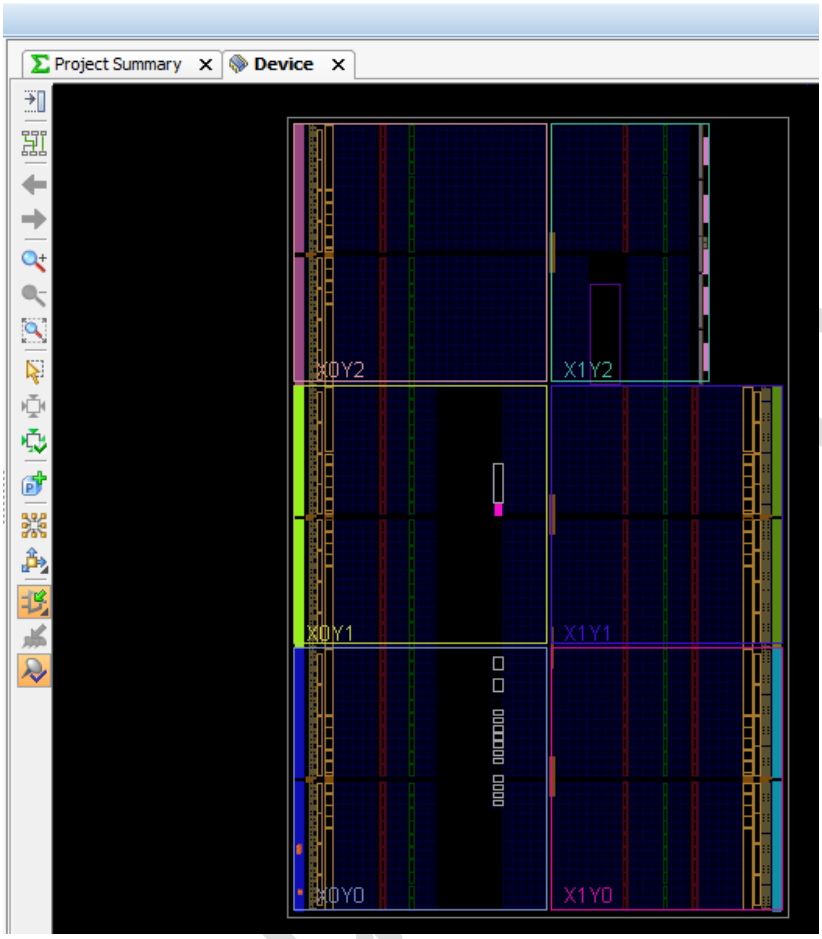


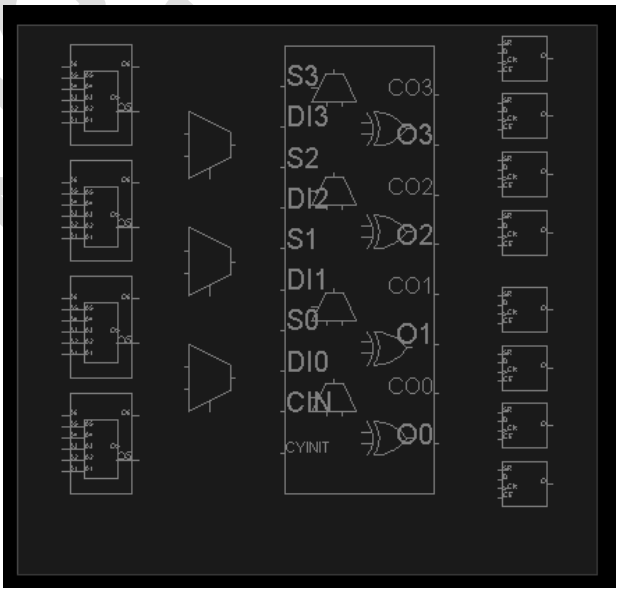
Fig. 25 - Opțiunea „Open Synthesized Design”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În această etapă, în cadrul mediului Vivado, se poate vizualiza o reprezentare grafică a conținutului capsulei FPGA, inculsiv porțiunea configurată în cadrul proiectului actual:



A.



B.

Fig. 26 – Reprezentarea grafică: A. - a capsulei FPGA, B. - A unui modul din cadrul capsulei

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- După cum se observă în cadrul figurii nr. 26, capsula unui FPGA, încorporează tot felul de componente logice (porți, bi-stabile, monostabile, memorii etc.) care nu sunt conectate între ele. Prin intermediul codului - program Verilog, se pot face asocieri și legături fizice între componentele logice. După cum era menționat în documentația anterioară, memoriile programabile sunte elementele de legătură dintre porți (ariile tranzistoarelor de legătură).

- În cadrul acestei etape de post – sitetizare pot fi realizate optimizări la nivelul schemei de implementare (vezi metodele de reducere a funcțiilor logice – Circuite Digitale anul II). În cazul de față fiind doar o poartă logică utilizată, etapa de optimizare post – sinteză poate fi omisă.

Următorul pas va fi implementarea schemei logice în cadrul capsulei FPGA în vederea generării fișierului „.bit” pentru configurare. În această etapă mediul Vivado, determină în mod automat cele mai optime rute pentru a creea legăturile dintre componente (metoda „place and route”. Procedeul poate fi inițiat prin intemrediul opțiunii „Run Implementation”. La finalizarea acestui proces va apărea o fereastră de dialog pentru confirmare.

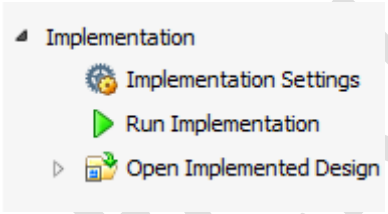


Fig. 27 – Opțiunea „Run Implementation” din bara de gestiune a proiectului

Pentru a genera fișierul de configurare „bit”, în cadrul ferestrei de confirmare nou deschise, se va alege opțiunea „Generate Bitsream”, și se va confirma comanda:

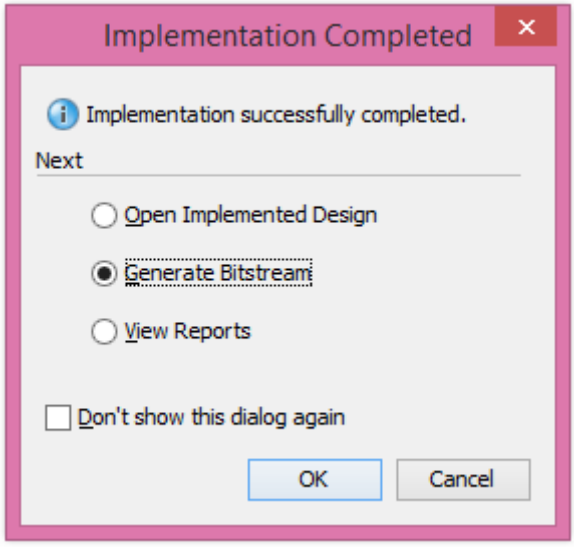


Fig. 28 – Opțiunea „Generate Bitstream”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Odată generat, fișierul executabil „.bit” poate fi implementat în memoria platformei FPGA Basys 3, prin intermediul adaptorului USB - jTAG de care dispune platforma Basys 3. Pentru a accesa programatorul platformei, se va deschide modulul „Hardware Manager” (modulul de gesiune al dispozitivelor fizice), prin intermediul opțiunii „Open Hardware Manager” din cadrul ferestrei de confirmare a generării fișierului executabil „.bit”:

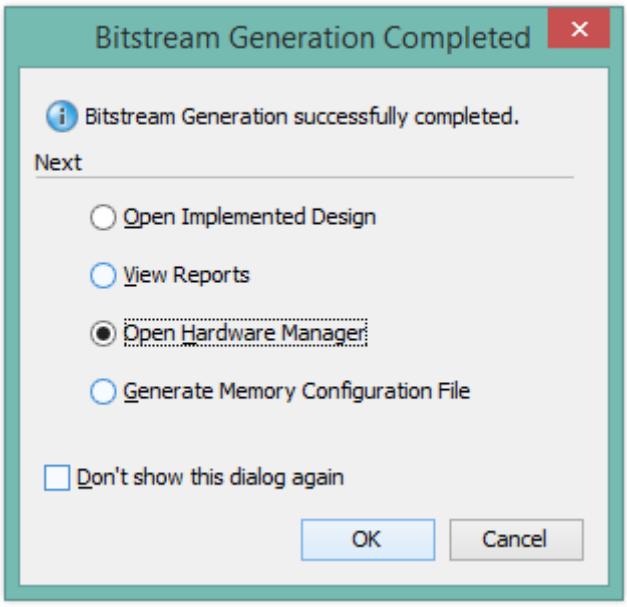


Fig. 29 – Opțiunea „Open Hardware Manager”

- Având platforma de dezvoltare Basys 3 conectată la portul USB al calculatorului, se va proceda la identificarea acestei prin intermediul opțiunii „Open Target și Auto Connect” din cadrul categoriei „Program and Debug”:

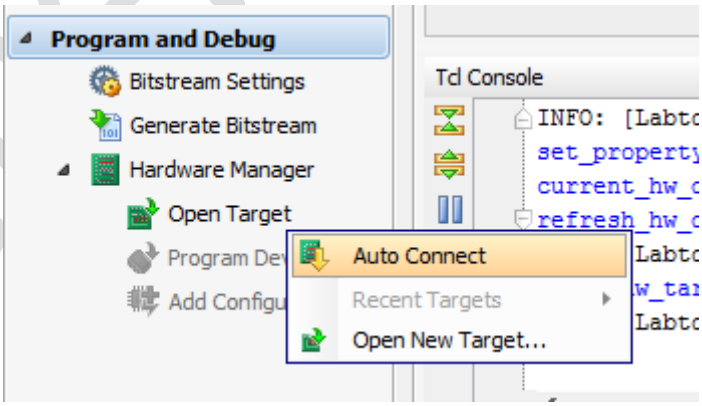


Fig. 30 – Opțiunea „Auto Connect”

- În urma conectării la platforma de dezvoltare Basys 3, opțiunea „Program Device” va deveni disponibilă. Cu ajutorul acestei opțiuni, fișierul executabil „.bit” va fi transferat în memoria platformei Basys 3. În urma alegerii acestei opțiuni, mediul Vivado va sugera tipul capsulei FPGA identificat, în cazul de față „xc7a35t_0”:

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

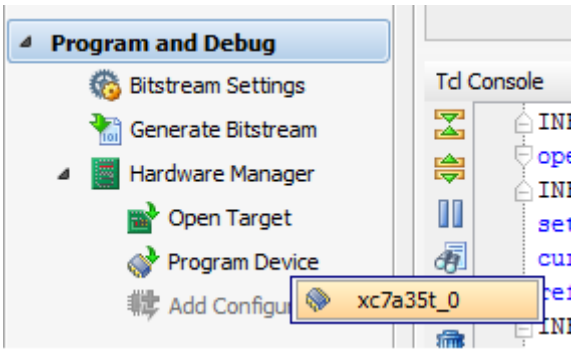


Fig. 31 – Opțiunea „Program Device” și identificarea capsulei FPGA

- În urma alegerii tipului de capsulă „FPGA”, mediul Vivado, va identifica locația (sau calea de acces) fișierului „.bit”. În cazul în care programarea se realizează prin intrermediul unui adaptor USB – jTAG extern, se va specifica tipul și configurația acestuia prin intermediul unui fișier special destinat configurării adaptorului (eng. Debug probe file). În urma identificării locației fișierului executabil „.bit”, se va alege opțiunea „Program”.

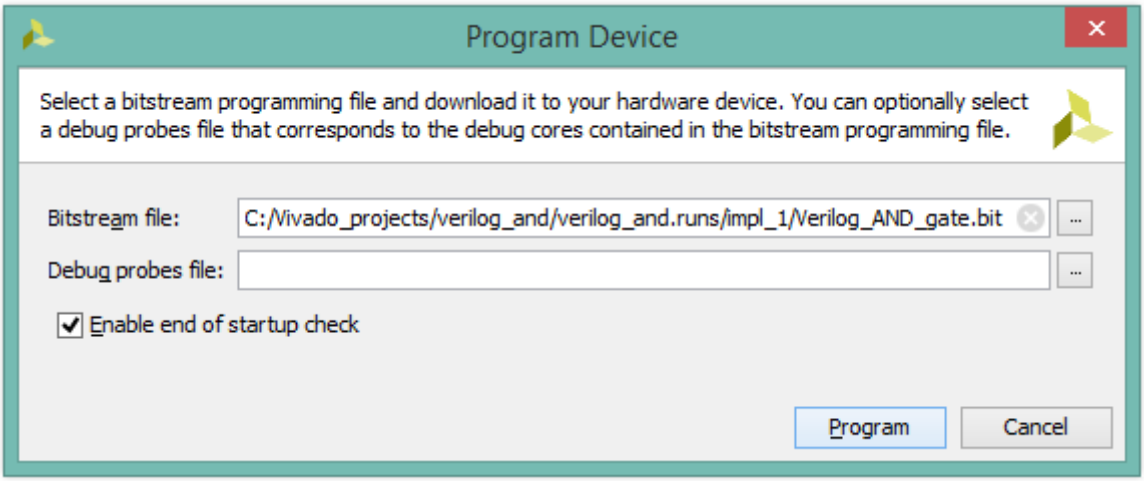
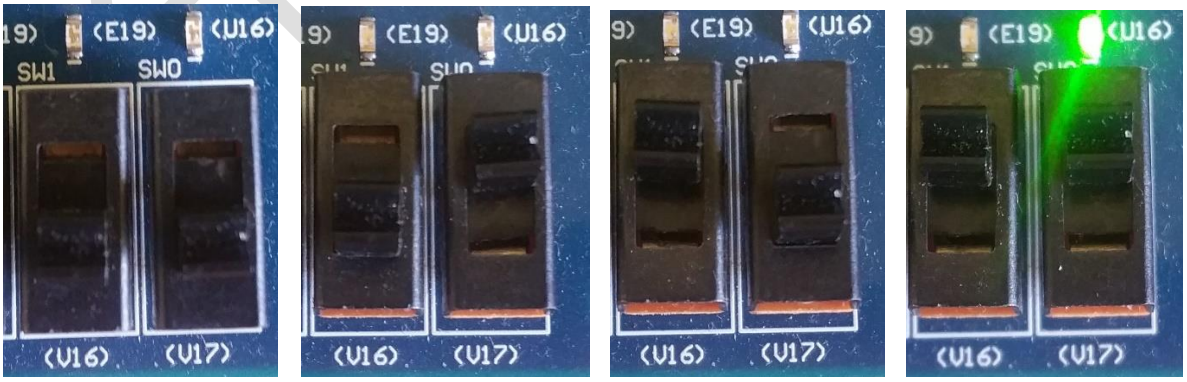


Fig. 32 – Identificarea locației fișierului „.bit” și opțiunea „Program”



A = 0; B = 0; f_AB = 0; A = 1; B = 0; f_AB = 0; A = 0; B = 1; f_AB = 0; A = 1; B = 1; f_AB = 1;

Fig. 33 – Testarea funcționalității circuitului logic implementat la nivel de FPGA

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

2. Implementarea unui circuit logic pe baza porții „ȘI” prin intermediul limbajului VHDL:

Acronimul VHDL mai precis VHSIC-HDL derivă de la denumirea (eng.) Very High Speed Integrated Circuit Hardware Description Language. Adică, este un limbaj pentru configurare fizică (eng. hardware) a circuitelor logice re-programabile, precum ariile de porți (eng. FPGA).

La fel ca și în cazul limbajului Verilog, **se va deschide mediul de programare și testare Vivado**, în cadrul căruia **se va iniția un proiect nou**. **Menționăm faptul că, etapele similare deja descrise în cadrul documentației actuale, nu vor mai fi prezentate în această secțiune, dar ele NU TREBUIE OMISE în vederea realizării proiectului!** Indiferent de faptul că abordarea codului program se realizează pe baza limbajului Verilog sau VHDL, etapele necesare pentru a programa sau re-configura o capsulă FPGA se parcurg în mod necesar, anume:

- definirea proiectului;
- definirea limbajului de redactare al fișierului sursă (VHDL sau Verilog);
- definirea modulelor sau entităților modulare;
- redactarea codului program în cadrul fișierului sursă;
- sintetizarea codului program;
- optimizarea rezultatului de sintetizare (etapa de post – sinteză);
- implementarea circuitului logic proiectat;
- generarea fișierului executabil „.bit”;
- încărcarea în memoria platformei a fișierului executabil de re-configurare „.bit”;

Astfel pentru a realiza un nou proiect, **se va alege denumirea acestuia ca și „VHDL_AND_gate”**. Pentru tipul de proiect **se va alege opțiunea „RTL Project”**. Ca și platformă de dezvoltare, **se va alege Basys 3** din categoria „Boards”. Procedura **se va finaliza prin alegerea opțiunii „Finish”.**

În cadrul noului proiect creeat (VHDL_AND_gate), **se va defini fișierul sursă**. În cazul de față, **se va alege tipul fișierului sursă „VHDL”**, **se va alege denumirea fișierului „VHDL_AND_gate”**, iar ca și locație sau apartenență **se va alege „Local to Project”.**

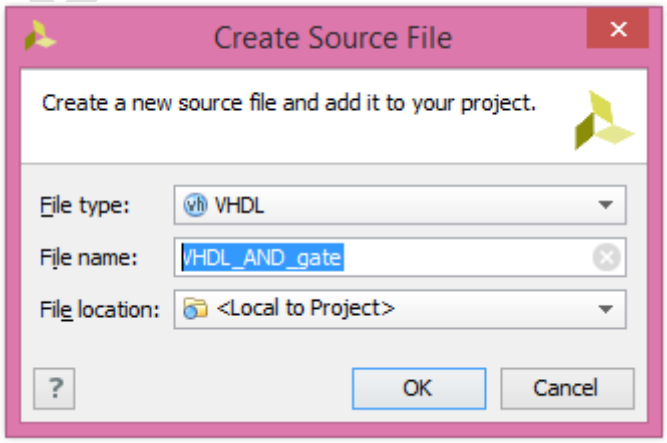


Fig. 34 – Definirea fișierului sursă VHDL

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În urma definirii fișierului sursă, **se vor defini și porturile de intrare sau ieșire ale entității modulare**, care se va creea în interiorul capsulei FPGA. În mod similar, ca și în cazul fișierului sursă „Verilog”, **se vor utiliza aceleași denumiri funcționale**:
- prima intrare: „A”;
- a doua intrare: „B”;
- singura ieșire: „f_AB”;

În urma definirii porturilor de intrare și ieșire, **se va creea un fișier nou de constrângere** specific proiectului. **Se va alege tipul fișierului „XDC”**. De asemenea **se va alege numele fișierului „vhdl_and_constraints”**, iar **locația sau apartenența va fi „Local to Project”**.

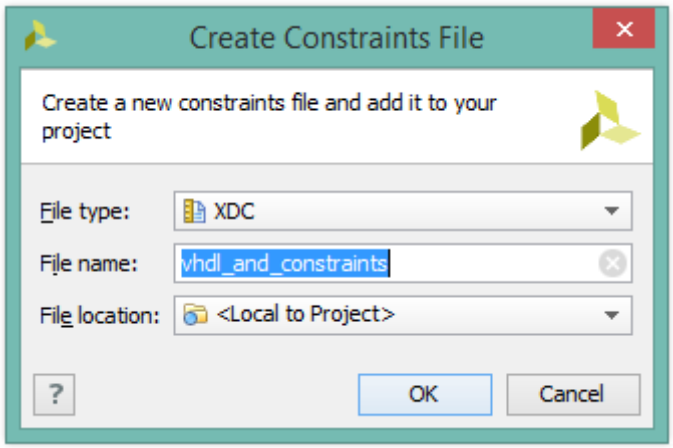


Fig. 35 – Definirea fișierului pentru constrângere

În urma definirii fișierului pentru constrângere, **se va proceda la redactarea codului – program** în limbaj VHDL. Acestă etapă poate fi realizată prin **editarea fișierului sursă „VHDL_AND_gate.vhd”.** Se vor introduce următoarele instrucțiuni în cadrul fișierului sursă:

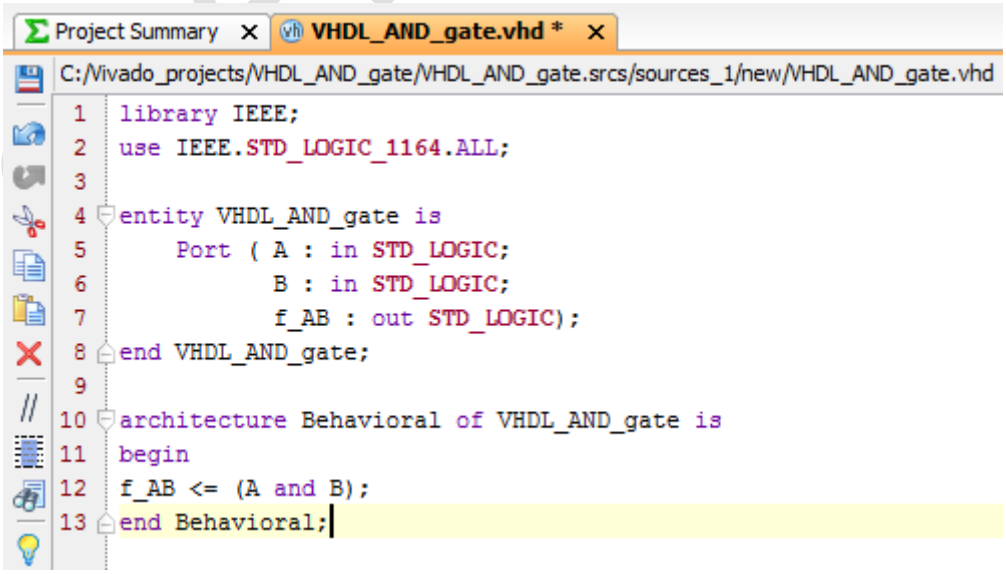


Fig. 36 – Editarea fișierului sursă „VHDL_AND_gate.vhd”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Spre deosebire de limbajul Verilog, în cadrul limbajului VHDL, structura logică elementară poartă denumirea de „entitate” modulară (eng. entity). De asemenea, în cadrul limbajului VHDL, se definește „comportamentul” entității logice (eng. behavioral) prin atribuirea către portul de ieșire a unei funcții logice definită pe baza porturilor de intrare. Simbolul „<=” reprezintă **instrucțiunea de atribuire**.

- În mod similar limbajului Verilog, și în cadrul limbajului VHDL se utilizează conceptul de „module logice”, având porturi de intrare și ieșire definite de către utilizator.

- În cazul de față codul - program VHDL scris realizează funcția „logic ȘI” (AND), astfel, descrie comportamentul unei porți fizice „ȘI”.

În urma definirii conținutului fișierului sursă, **se va proceda la definirea conținutului fișierului de constrângere**. În mod similar, ca și în cazul limbajului Verilog, fișierul de constrângere va avea **același conținut**. Adică, intrarea „A” va fi atribuită comutatorului „V17”, intrarea „B” comutatorului „V16”, iar ieșirea „f_AB” va fi atribuită indicatorului luminos „U16”.

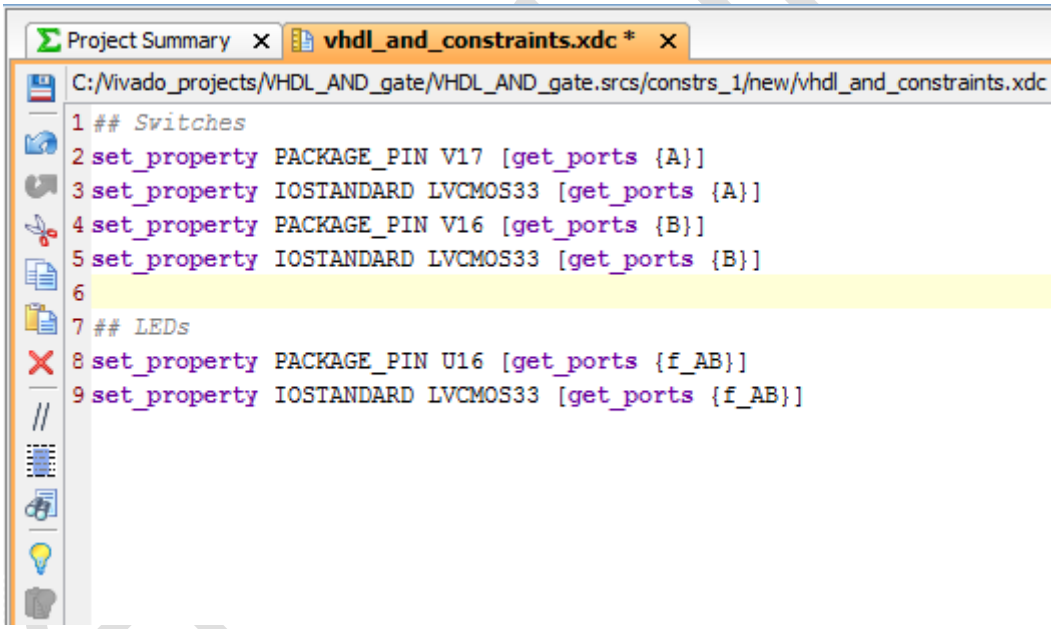


Fig. 37 – Editarea fișierului de constrângere

În urma stabilirii constrângerilor, și a conținutului fișierului sursă VHDL, **se va proceda la etapa de sintetizare a codului VHDL**. Inițierea procesului de sintetizare se va realiza prin intermediul opțiunii **„Run Synthesis”** din cadrul ferestrei de gestiune a proiectului.

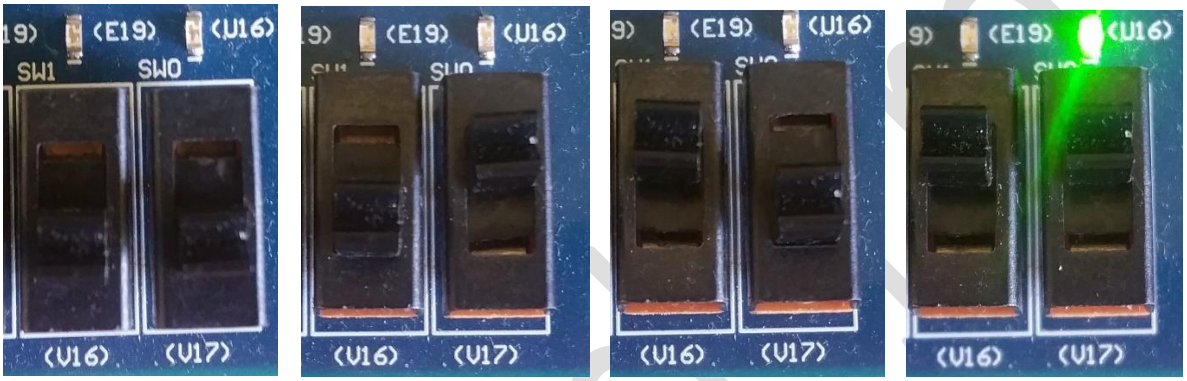
Mesajul de confirmare a finalizării procesului de sinteză, pune la dispoziție mai multe opțiuni printre care și opțiunea de implementare. **Se va proceda la etapa de implementare** prin alegerea opțiunii **„Run Implementation”** în cadrul mesajului de confirmare.

După etapa de implementare, **se va proceda la etapa de generare a fișierului executabil binar** „.bit” prin intermediul opțiunii **„Generate Bitstream”** din cadrul mesajului de confirmare a finalizării etapei de implementare.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

În cadrul mesajului final de confirmare a finalizării etapei de generare a fișierului „bit”, **se va alege opțiunea „Open Hardware Manager”.** În cadrul ferestrei pentru gestiune a proiectului **se va alege opțiunea „Open Target”** și **„Auto Connect”.** Pentru a încărca fișierul executabil „.bit” generat, în memoria platformei Basys 3, **se va alege opțiunea „Program Device”** din cadrul ferestrei pentru gestiune a proiectului, apoi **se va alege tipul capsulei FPGA „xc7a35t_0”.** În urma acestei operații se va deschide fereastra de localizare a fișierului executabil „.bit”. **Se va stabili calea de acces înspre fișierul generat**, după care **se va alege opțiunea „Program”** pentru a încărca fișierul „.bit” în memoria platformei Basys 3.

Pentru a testa funcționalitatea circuitului logic implementat la nivel capsulei FPGA, se vor realiza toate combinațiile tabelului de adevăr al unei porți „ȘI” (AND).



A = 0; B = 0; f_AB = 0; A = 1; B = 0; f_AB = 0; A = 0; B = 1; f_AB = 0; A = 1; B = 1; f_AB = 1;

Fig. 38 – Testarea funcționalității circuitului logic implementat la nivel de FPGA

3. Implementarea unui circuit logic pe baza porții „ȘI” (AND) prin intermediul generatorului automat de cod System Generator împreună cu mediul Matlab - Simulink:

Pe lângă mediul de dezvoltare Vivado, în cadrul pachetului Xilinx Vivado Design Suite – System Edition, mai există și generatorul automat de cod Xilinx System Generator sau Xilinx Model Composer ambele generatoare dezvoltate în exclusivitate pentru Matlab – Simulink.

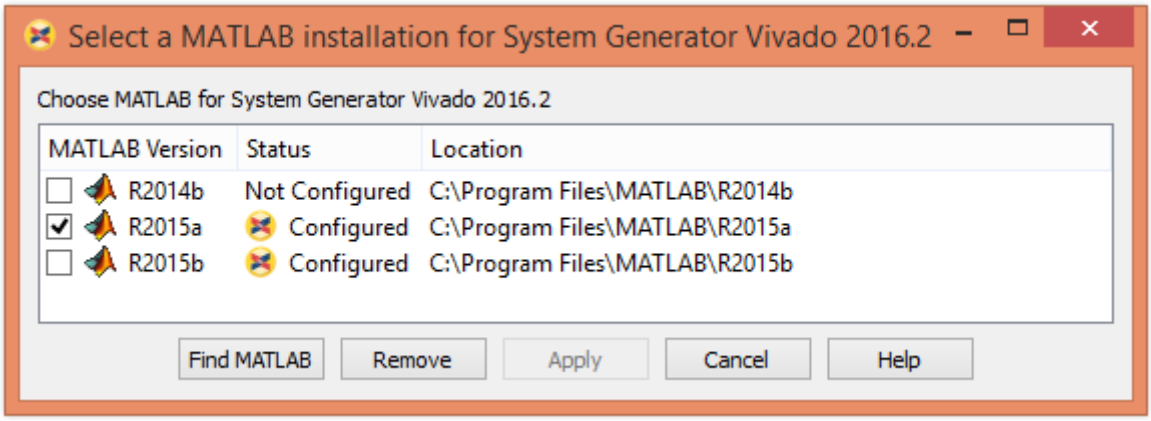


Fig. 39 – Configurarea generatorului automat de cod System Generator

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Odată integrat cu versiunea compatibilă de Matlab – Simulink, generatorul automat de cod, va pune la dispoziția utilizatorului o paletă vastă de instrumente specializată pentru dezvoltarea codului - program sub formă de model Matlab – Simulink.

IMPORTANT! Din motive de compatibilitate, pentr platforma Basys 3, se preferă versiunea Vivado 2016.2 – System Generator, integrat cu versiunea R2015a a mediului Matlab – Simulink. De asemenea, pentru a introduce platforme suplimentare în lista de co-simulare a generatorului automat de cod, trebuie editat fișierul „startup.m" în care se vor adăuga toate căile de acces înspre directorul cu fișierele de bază ale platformelor respective (Digilent Boards Support Package).

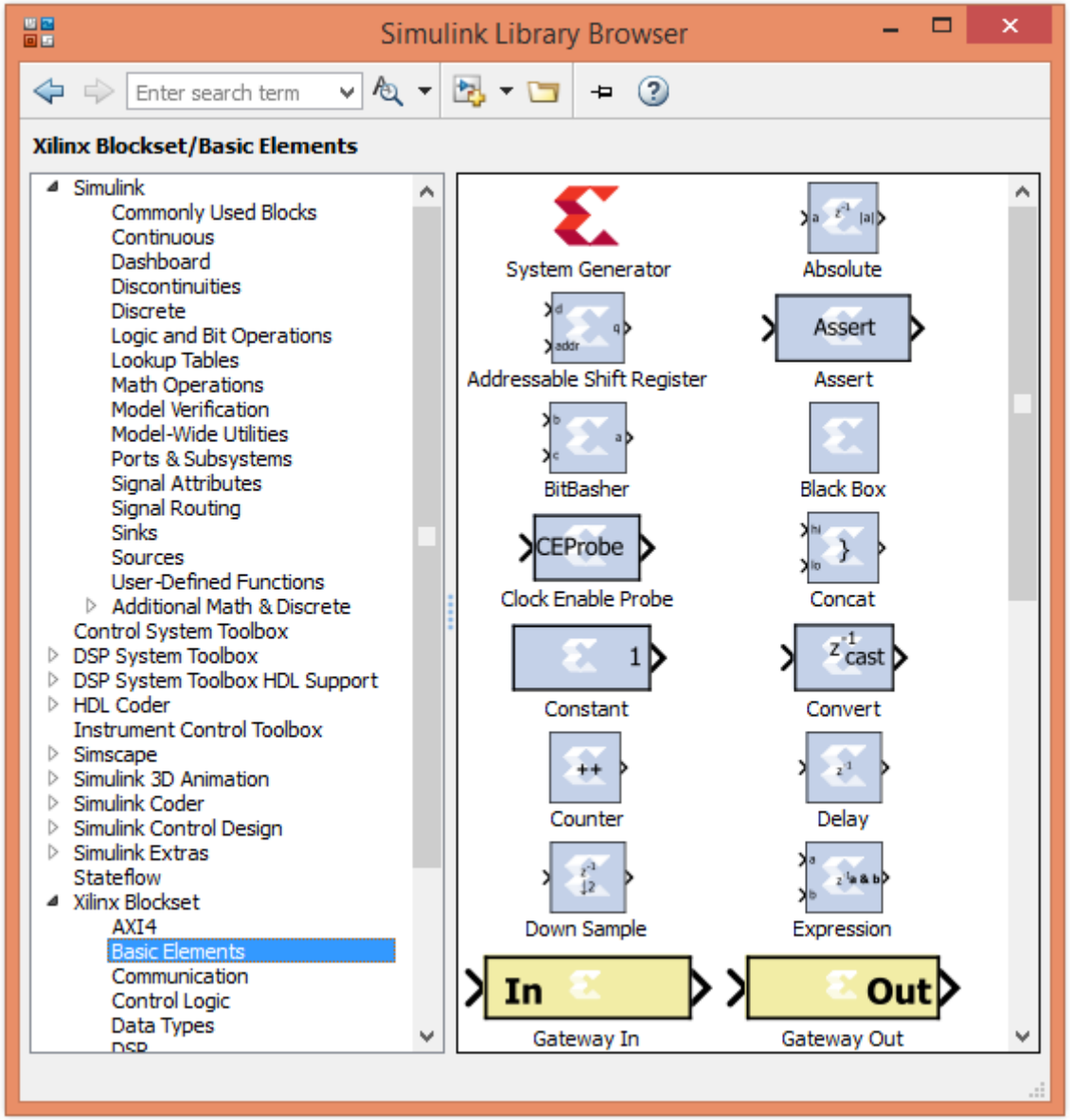


Fig. 40 – Paleta de instrumente a generatorului automat de cod

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Instrumentele din cadrul acestei palete sunt disponibile doar la lansarea în execuție a mediului Matlab – Simulink prin intermediul generatorului de cod System Generator.

Pentru început, **se va iniția un model Matlab – Simulink nou**, și **se va salva sub** denumirea „sysgen_AND_gate**”**. În cadrul modelului **se vor realiza următoarele parametrizări inițiale** din cadrul meniului „Simulation”, opțiunea „Model Configuration Parameters”:
- Stop time = inf;
- Solver options – Type = Fixed-step;
- Solver = discrete (no continuous states);
- Fixed-step size (fundamental sample time) = 1e-4;
- Se vor confirma toate parametrizările prin apăsarea butonului „Apply” apoi „Ok”.

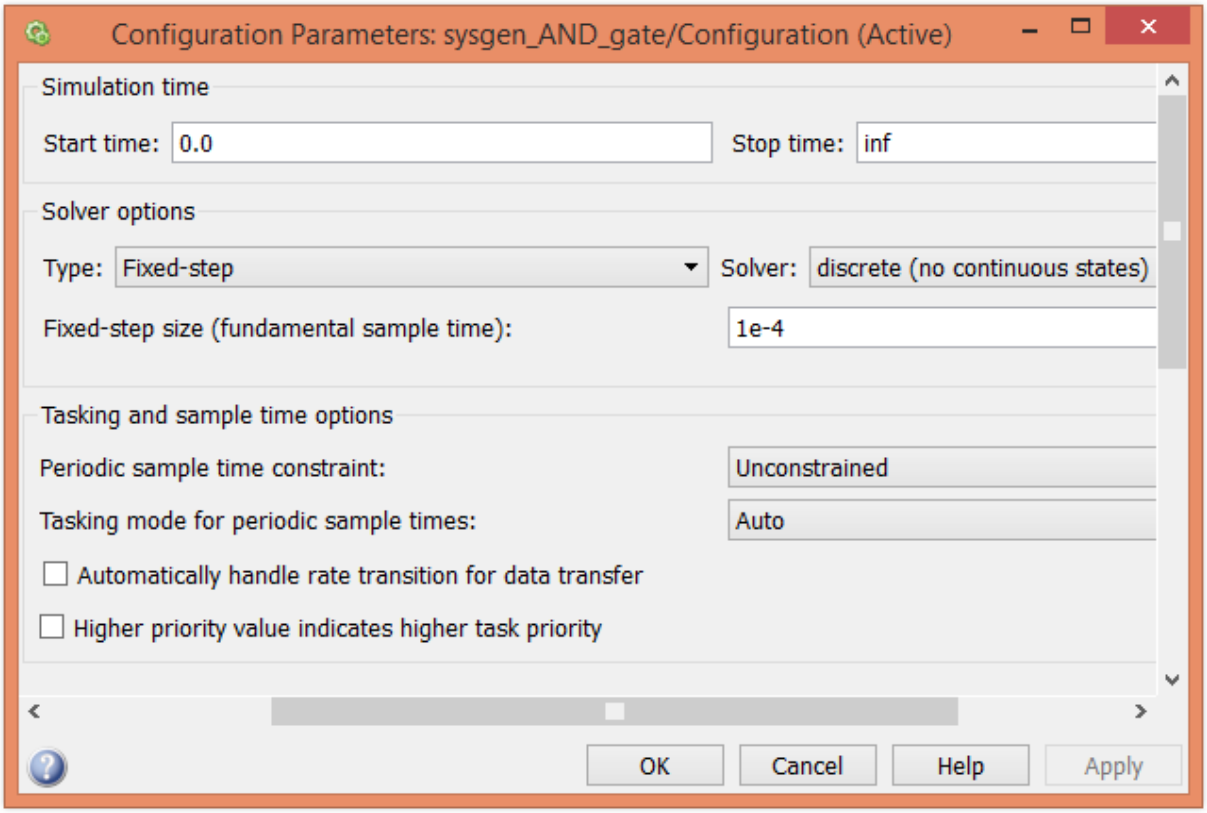


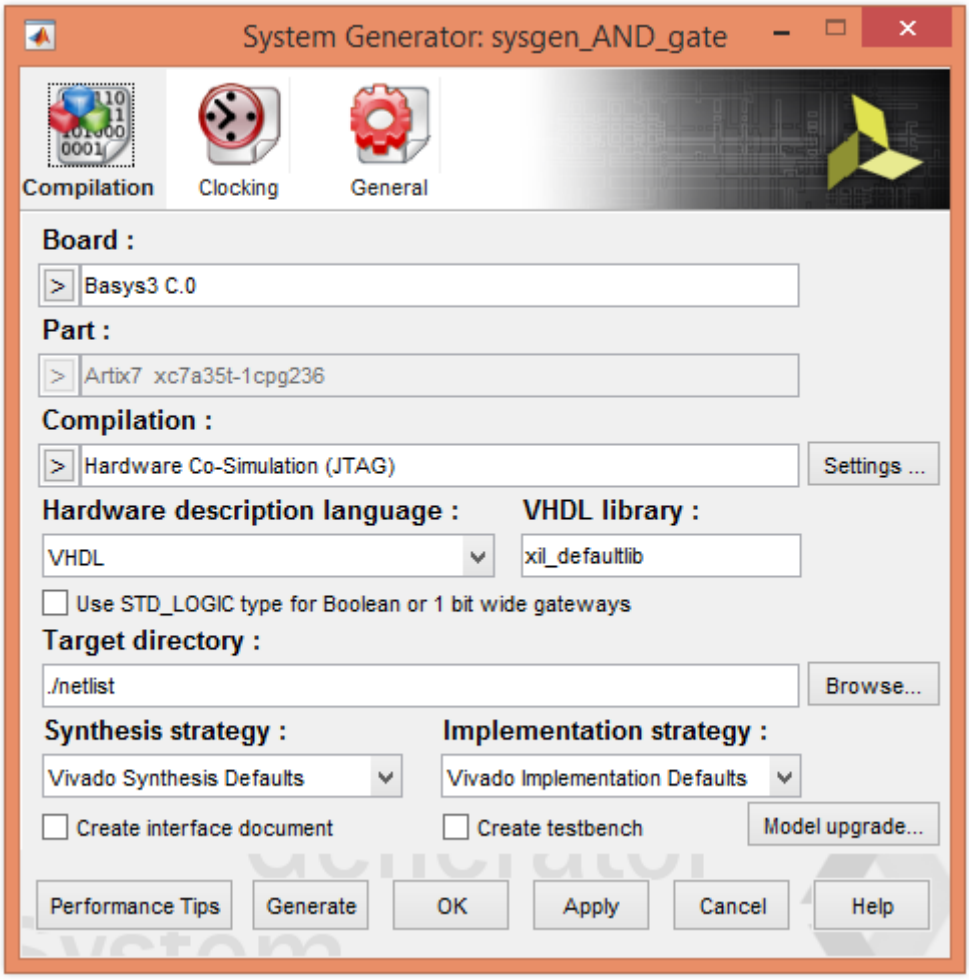
Fig. 41 – Parametrizarea inițială a modelului Matlab – Simulink

În cadrul modelului **se va introduce blocul specific generatorului de cod** System Generator, prin intermediul căruia **se vor realiza parametrizările specifice platformei FPGA**:

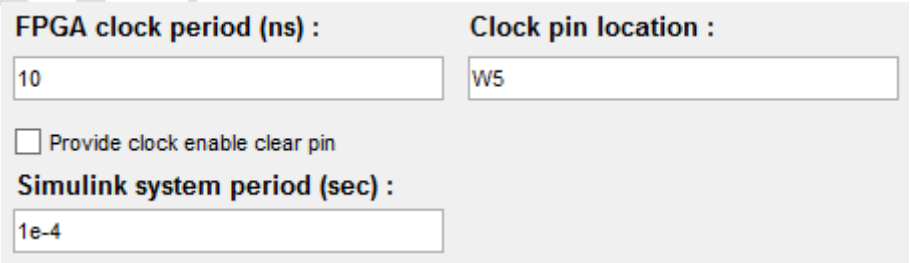
Fig. 42 – Blocul specific generatorului de cod System Generator

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Parametrizările specifice platformei FPGA sunt:
În categoria „Compilation”:
- Board = Basys3 C.0;
- Compilation = Hardware Co-Simulation (JTAG);
În categoria „Clocking”:
- Clock pin location = W5;
- Simulink System period (sec) = 1e-4;



A.



B.

Fig. 43 – Parametrizările specifice platformei FPGA Basys 3

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pentru platforma FPGA **se va implementa următorul model Simulink**:

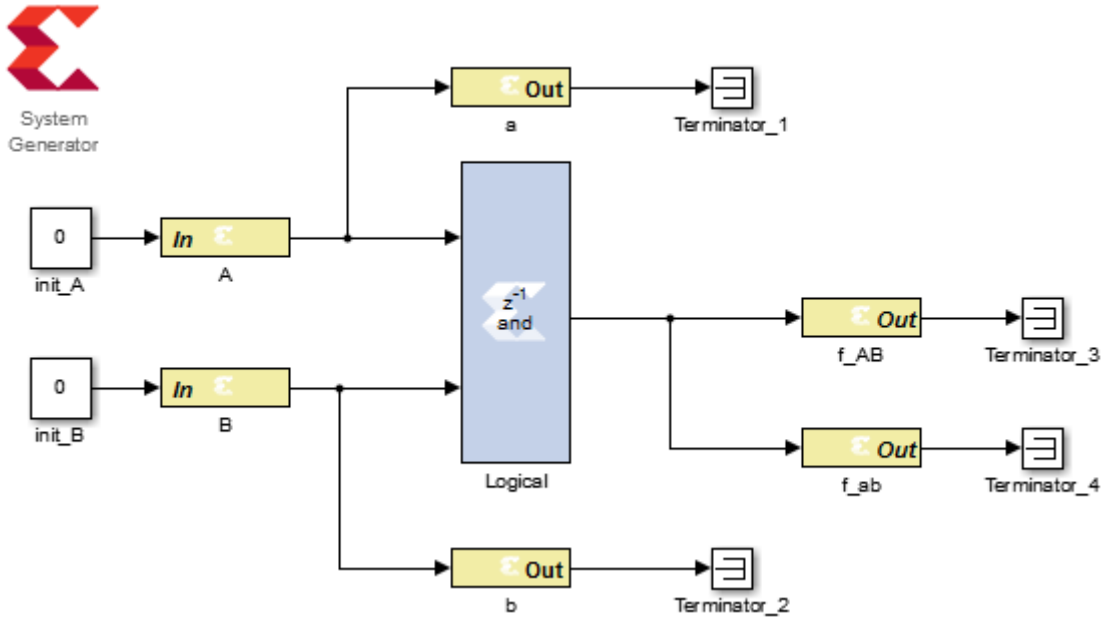


Fig. 44 – Modelul Simulink al codului - program pentru platforma FPGA Basys 3

Simbol bloc	Denumire bloc	Categorie / Subcategorie
System Generator	System Generator	Xilinx Blockset – Basic Elements
0 init_A	Constant	Simulink – Sources
In A	Gateway In	Xilinx Blockset – Basic Elements
Out f_AB	Gateway Out	Xilinx Blockset – Basic Elements
Terminator_1	Terminator	Simulink – Sinks
z-1 and AND	Logical	Xilinx Blockset – Control Logic

Pentru a dezvolta o aplicație de co-simulare în timp real pentru platforma FPGA în cadrul mediului Matlab – Simulink, este necesară implementarea unui model atât pentru codul - program al platformei cât și unul pentru calculatorul gazdă. Canalul de comunicație se stabilește prin intermediul protocolului jTAG – Real – Time Data eXchange (RTDX).

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

- Rolul modelului dezvoltat pentru platforma FPGA este de a permite generarea fișierului executabil „.bit” care urmează a fi încărcat și executat în memoria platformei.

Pentru a implementa un circuit logic pe baza unei singure porți „ȘI” (AND), se va realiza modelul indicat în fig. 44, iar în cadrul următoarelor blocuri se vor aplica parametrizări precum:
- În cadrul blocului „Gateway IN” (denumit „A”), în categoria „Basic”, **se va stabili tipul de date** (eng. Output Type) „Boolean”, și **timpul de eșantionare** (eng. Sample Period) „1e-4”;
- Pentru a impune constrângerile necesare, în cadrul aceluiaș bloc, în categoria „Implementation” **se va bifa opțiunea** „Specify IOB location constraints”, iar în cadrul căsuței denumită „IOB pad locations” **se va trece** „{'V17'}” iar în cadrul căsuței denumită „IO Standards” **se va trece** „{'LVCMOS33'}”. Pentru opțiunea „Interface” **se va alege** „None”;

- În cadrul blocului „Gateway IN” (denumit „B”), în categoria „Basic”, **se va stabili tipul de date** (eng. Output Type) „Boolean”, și **timpul de eșantionare** (eng. Sample Period) „1e-4”;
- Pentru a impune constrângerile necesare, în cadrul aceluiaș bloc, în categoria „Implementation” **se va bifa opțiunea** „Specify IOB location constraints”, iar în cadrul căsuței denumită „IOB pad locations” **se va trece** „{'V16'}” iar în cadrul căsuței denumită „IO Standards” **se va trece** „{'LVCMOS33'}”. Pentru opțiunea „Interface” **se va alege** „None”;

Pentru a prelua în calculatorul gazdă informația asupra stării intrărilor (sau formele de undă), **se vor introduce** două blocuri de ieșire de tip „Gateway Out”, denumite „a” și „b”, iar la ieșirea acestora, se vor adăuga blocuri terminatoare de capăt „Terminator”. În cadrul acestor blocuri **nu se vor impune constrângeri fizice** (eng. hardware).

În cadrul blocului „Gateway Out” (denumit „f_AB”), în categoria „Implementation” **se va bifa opțiunea** „Specify IOB location constraints”, iar în cadrul căsuței denumită „IOB pad locations” **se va trece** „{'U16'}” iar în cadrul căsuței denumită „IO Standards” **se va trece** „{'LVCMOS33'}”. Pentru opțiunea „Interface” **se va alege** „None”;

De asemenea, pentru a prelua în calculatorul gazdă informația asupra stării ieșirii funcției logice, **se va introduce** blocul „Gateway Out” (denumit „f_ab”) în cadrul căruia **nu se vor impune** constrângeri fizice (eng. hardware). Ieșirile se vor lega la blocuri terminatoare.

Blocul logic utilizat în cadrul modelului implementează funcția „ȘI” (AND), având comportamentul similar al unei porți logice de tip „ȘI” (AND). Practic în interiorul capsulei FPGA, se va configura în mod fizic o poartă logică de tip „ȘI” (AND), în legătură cu două comutatoare la intrările „A” și „B” și un indicator luminos la ieșirea „f_AB”.

Blocurile „init_A” și „init_B” reprezintă valorile constante prin intermediul cărora se stabilesc condițiile inițiale date circuitului logic la intrările „A” și „B”.

Blocurile „Terminator” au rolul de a încheia traseul fluxului informațional din cadrul modelului.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

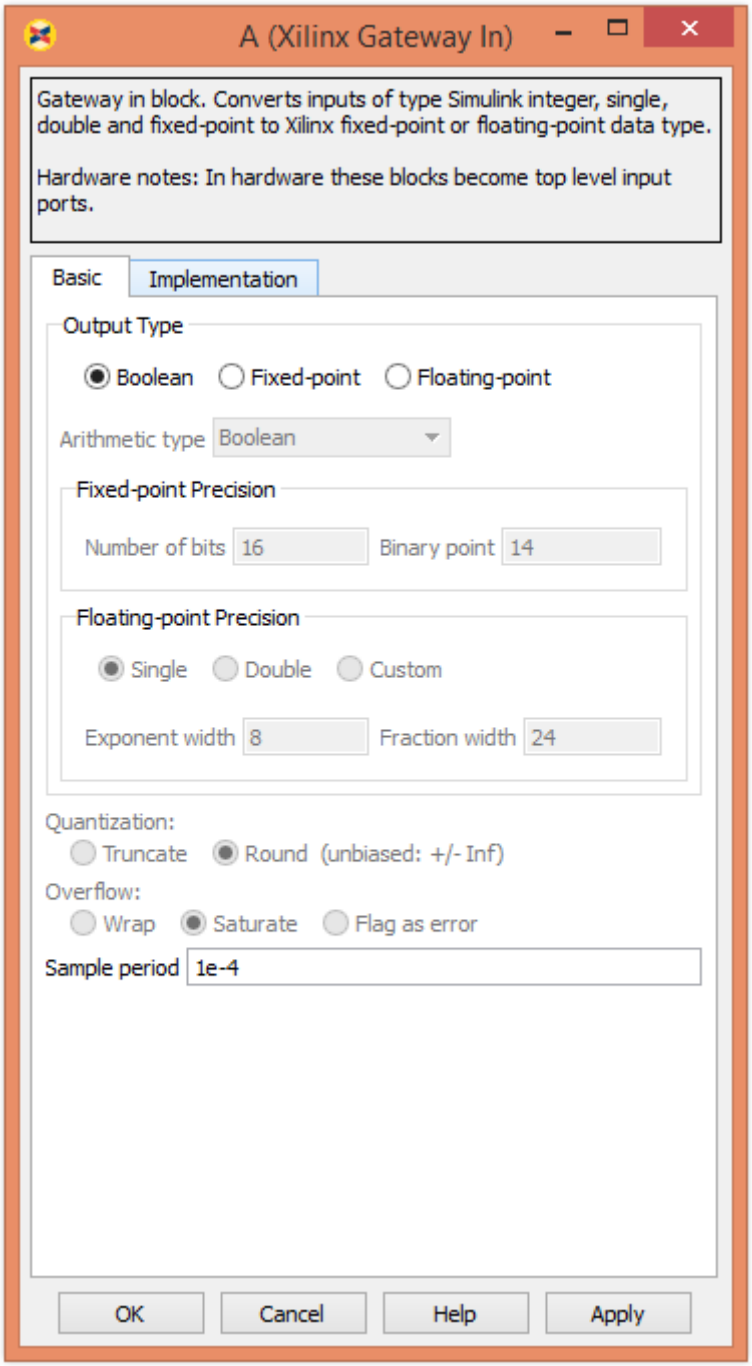


Fig. 45 – Configurarea blocului „Gateway In” categoria „Basic”

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

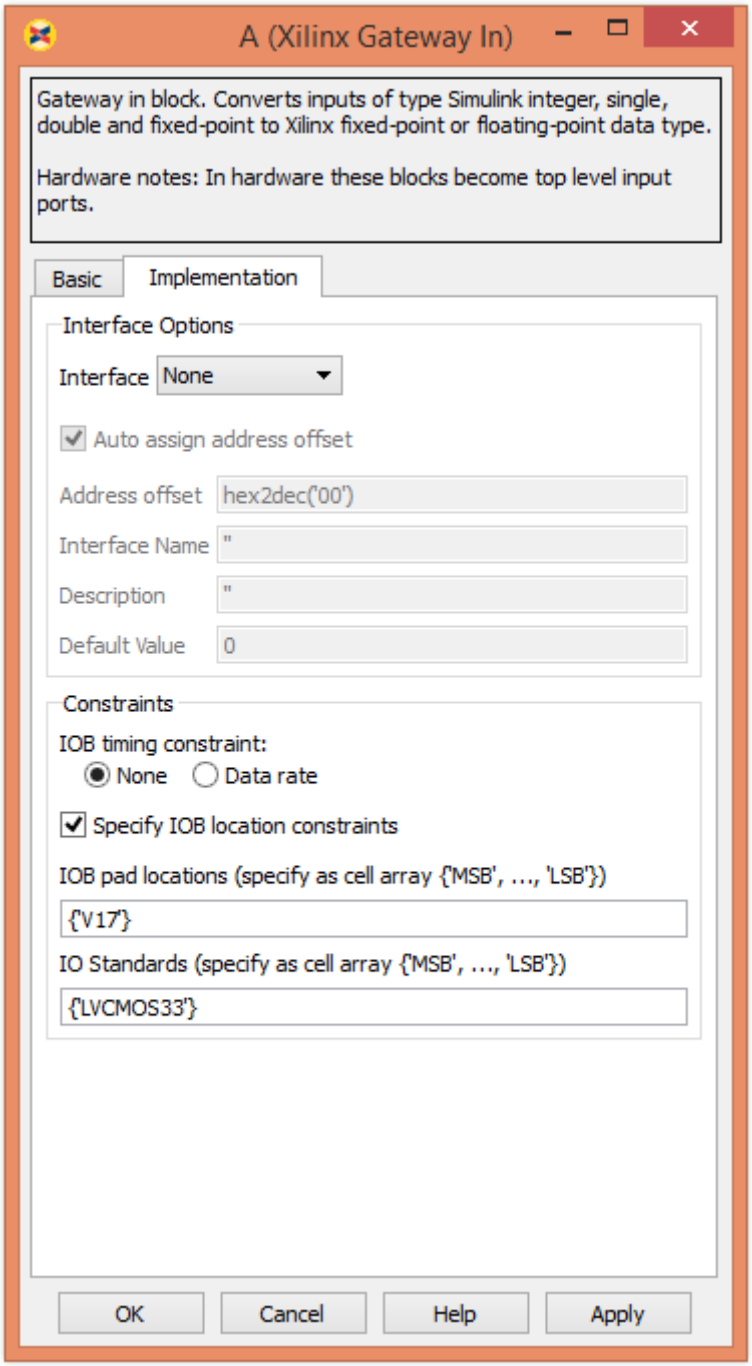


Fig. 46 – Configurarea blocului „Gateway In” categoria „Implementation”

- În vederea impunerii constrângerilor pentru intrarea „B” și pentru ieșirea „f_AB” ale porții logice se va proceda în mod similar intrării „A” (precum în figura nr. 45 și 46).

În urma impunerii parametrizărilor indicate în cadrul modelului specific codului – program, se va proceda la pasul următor, anume, la pasul de sintetizare, implementare și generare a fișierului „bit”. În cazul mediului Matlab – Simulink, în urma procesului de sinteză și generare, se va creea în mod automat o bibleotecă specifică proiectului în cadrul căreia, va fi inclus blocul de comunicare al programului executat în cadrul platformei FPGA.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pentru a iniția procesul de sintetizare, implementare și generare automată de cod, în cadrul blocului „System Generator” **se va alege opțiunea** „Generate”:

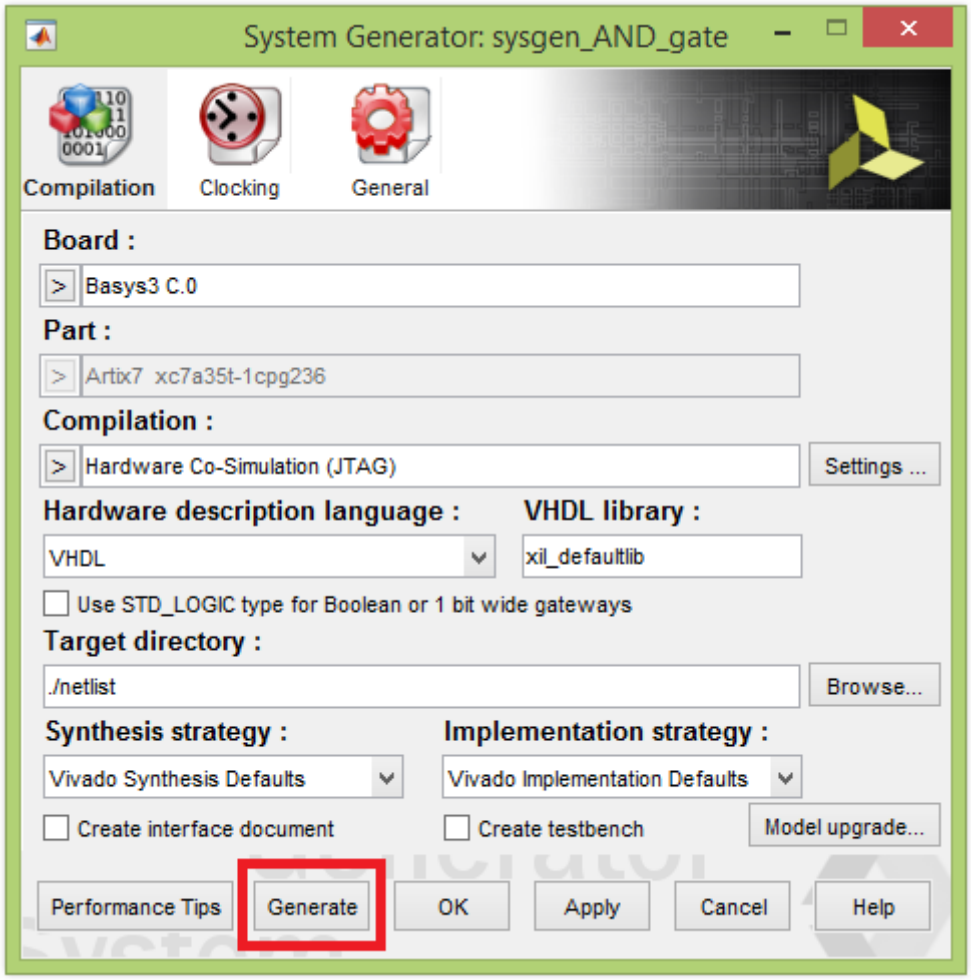


Fig. 47 – Generarea automată a codului program

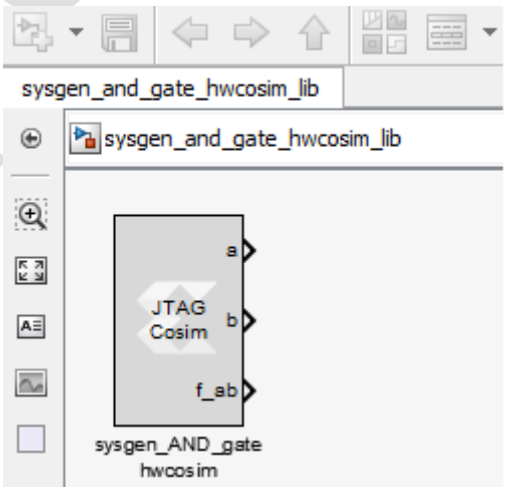


Fig. 48 – Blocul de comunicație între platforma FPGA și calculatorul gazdă

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pentru a examina comportamentul codului – program, la nivelul calculatorului gazdă, **se va concepe următorul model Simulink**, pe baza blocului de comunicație rezultant:

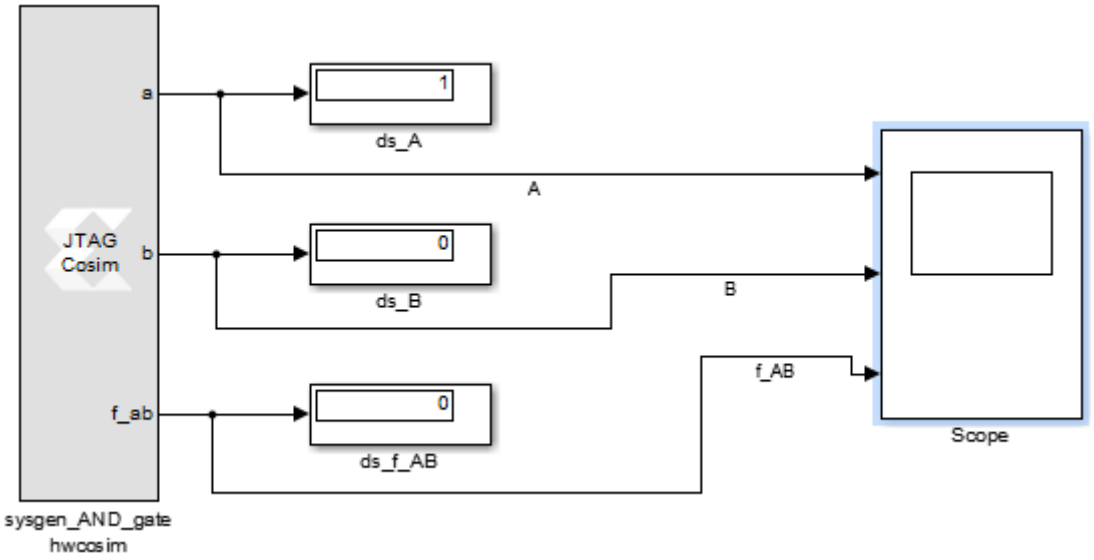
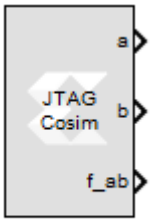
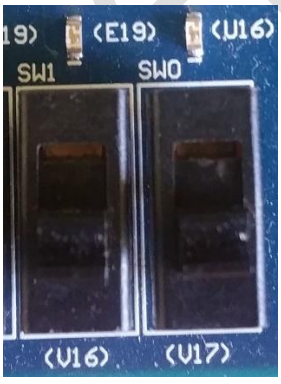
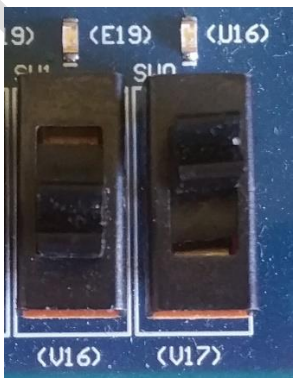


Fig. 49 – Modelul Simulink pentru calculatorul gazdă

Simbol bloc	Denumire bloc	Categorie / Subcategorie
	Scope	Simulink – Sinks
	Display	Simulink – Sinks
	Hardware co-simulation	Bibleoteca rezultantă în urma procesului de sintetizare și generare automată de cod

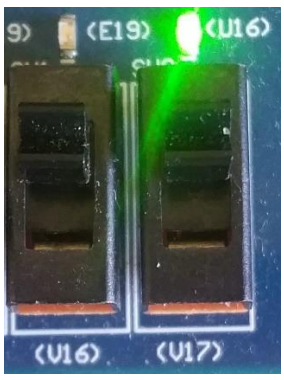


A = 0; B = 0; f_AB = 0;



A = 1; B = 0; f_AB = 0;

A = 0; B = 1; f_AB = 0;



A = 1; B = 1; f_AB = 1;

Fig. 50 – Testarea funcționalității circuitului logic implementat la nivel de FPGA

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Funcționarea modelului va putea fi examinată atât prin intermediul semnalizării indicatorului luminos, cât și prin intermediul afișajelor din model (eng. Display), dar și prin intermediul osciloscopului virtual implementat în cadrul modelului. Pentru a pune în funcțiune circuitul logic implementat la nivelul capsulei FPGA, **se va alege opțiunea** „Run” din cadrul meniului „Simulation”. În momentul în care caclulatorul gazdă va încărca fișierul executabil în memoria platformei, modelul de pe calculatorul gazdă va afișa rezultatele co-simulării în timp real atât pe afișajele din model cât și pe osciloscop, în funcție de cum sunt acționate comutatoarele specificate prin constrângeri. În momentul în care ambele comutatoare se vor afla în stare activă, toate afișajele din model vor arăta valoarea „1” iar indicatorul luminos verde (U16) din cadrul platformei va semnaliza. În cadrul osciloscopului virtual, se vor putea observa formele de undă ale semnalelor logice de intrare („A” și „B”), dar și semnalul de ieșire al porții logice „ȘI” (AND). Se observă faptul că, în momentul în care ambele semnale de intrare sunt în stare activă („logic 1”) și semnalul de la ieșire trece în stare activă (conform tabelului de adevăr specific comportamentului logic al unei porți „ȘI”).

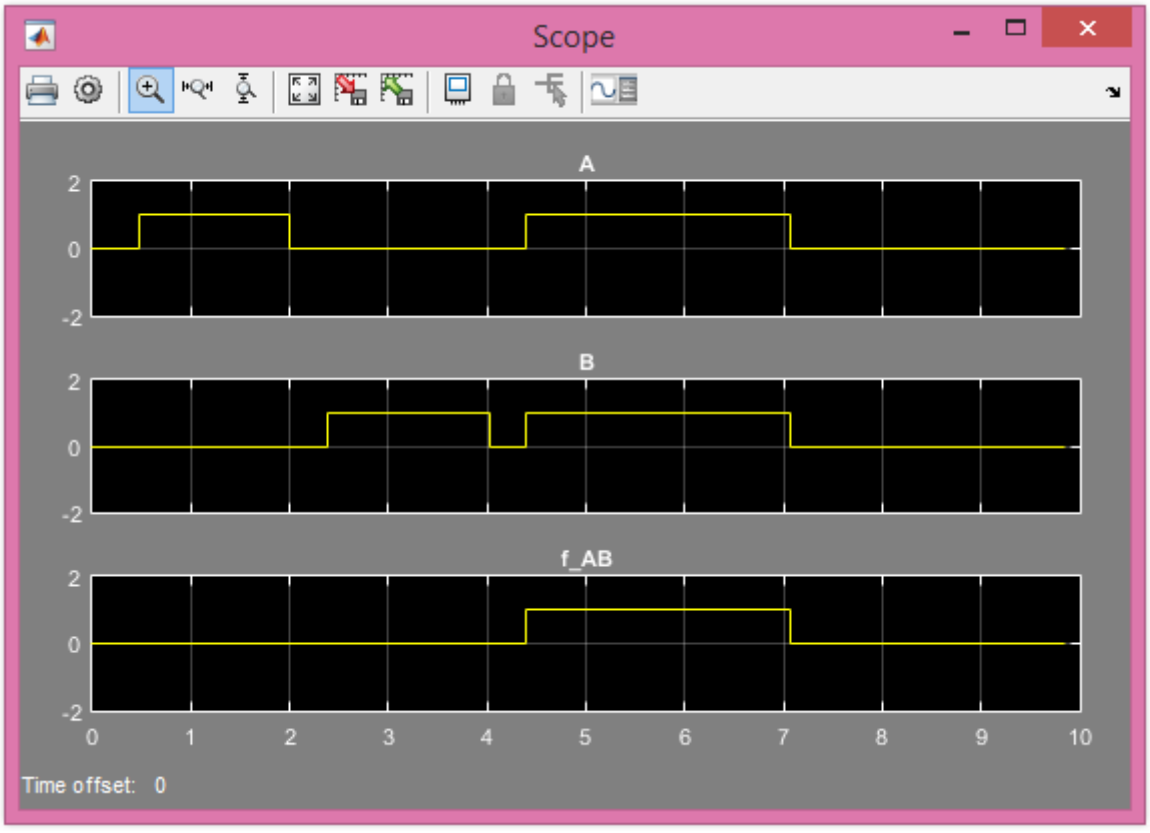


Fig. 51 – Afișarea stărilor logice ale intrărilor și ieșirilor specifice circuitului implementat

III. PROGRAMAREA PLATFORMEI DE DEZVOLTARE ZEDBOARD – FPGA ZYNQ– 7000: [7] [8]

Metodele anterioare de programarea amintite pentru platforma BASYS 3 – sunt compatibile și cu platforma ZedBoard. Diferența majoră, constă în faptul că, în mediul Matlab – Simulink, generarea automată de cod poate fi realizată printr-o altă metodă decât System Generator + Matlab Simulink, anume, prin metoda **HDL Coder + Matlab Simulink**. Pachetul de

instrumente HDL Coder, este o extensie pentru mediul de simulare Matlab – Simulink, care îi permite mediului de simulare, să genereze codul program care poate fi încărcat în memoria platformei de dezvoltare cu FPGA. Tot în cazul platformei ZedBoard, se poate utiliza de asemenea și metoda de programare Matlab – Simulink + System Generator.

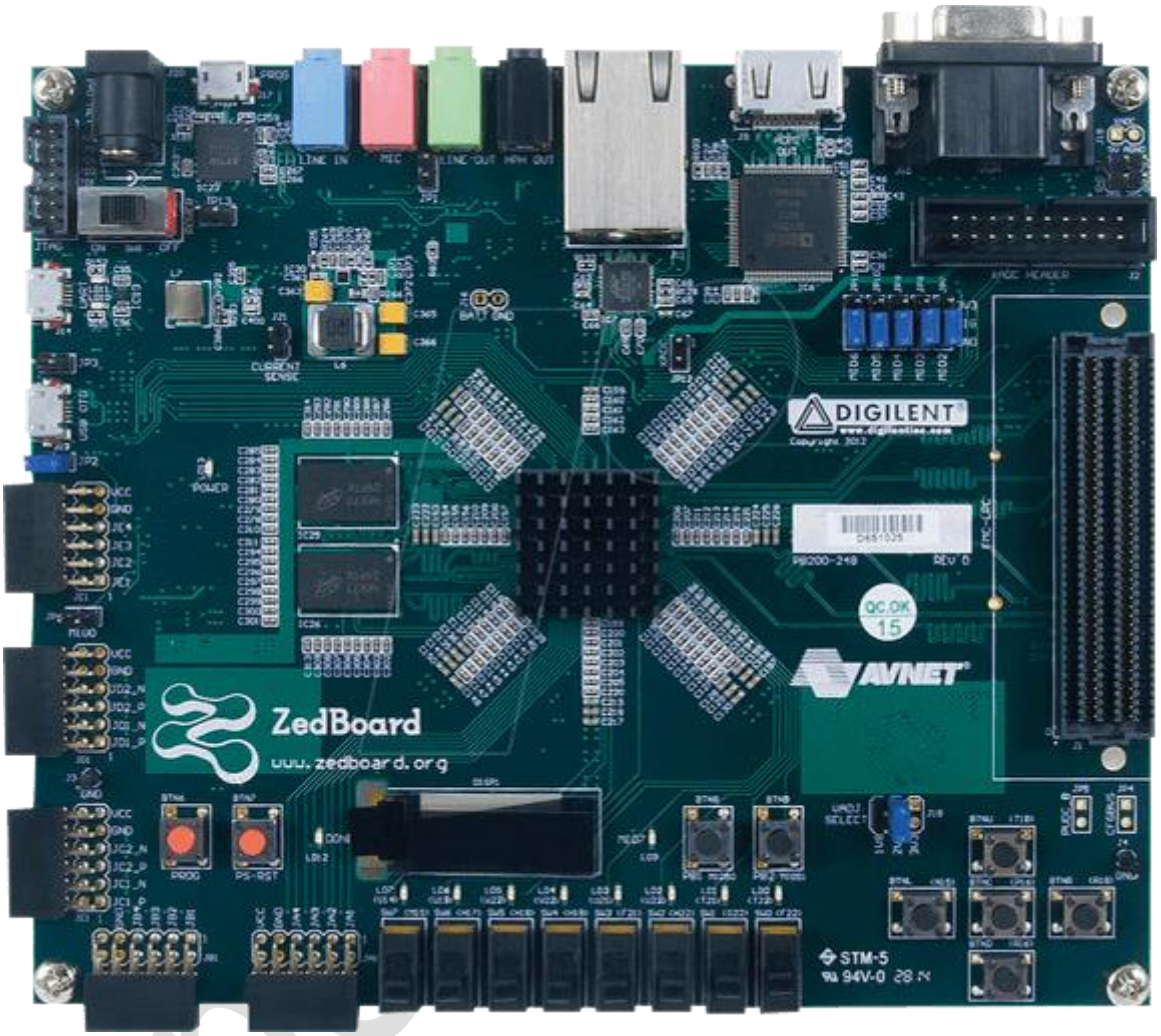


Fig. 52 – Platforma de dezvoltare cu FPGA ZedBoard – Zynq 7000 [6]

Pentru a realiza operația de programare și configurare a ariei de porți din cadrul platformei ZedBoard, pe baza mediului de simulare Matlab – Simulink și a pachetului de instrumente HDL Coder, sunt necesare următoarele programe și extensii:

- Matlab R2018b cu pachetele de instrumente: HDL Coder și Embedded Coder;
- Xilinx Vivado 2016.2 – System Edition;
- Pachetul de suport pentru platformele Digilent în cadrul mediului Xilinx Vivado;
- Pachetele de suport pentru platformele FPGA Zynq-7000 în mediului Matlab – Simulink;
- Pachetele de suport pentru platformele SoC – ARM – Zynq – 7000 în Matlab – Simulink;

În vederea generării codului – program pe baza mediului Matlab – Simulink, se vor utiliza cele două generatoare automate de cod Embedded Coder pentru procesorul ARM și HDL Coder pentru nucleul FPGA Zynq – 7000. Pentru a programa platforma ZedBoard cu

ajutorul mediului de simulare testare și programare Matlab – Simulink, se vor instala pachetele de suport „Embedded Coder Support Package for Xilinx Zynq Platform” (pentru programarea procesorului ARM) și „HDL Coder Support Package for Xilinx Zynq Platform” (pentru programarea ariei de porți eng. FPGA).

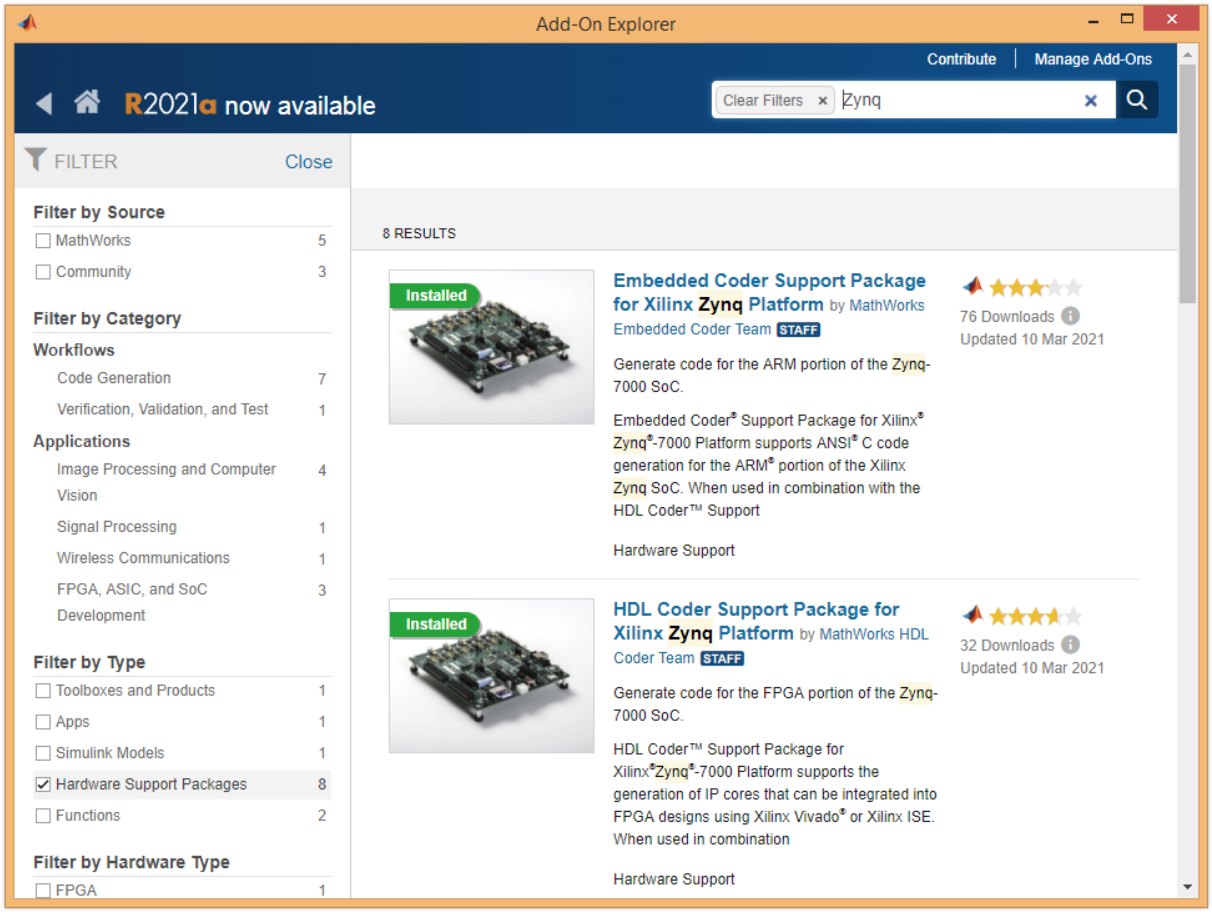


Fig. 53 – Pachetele de suport Matlab – Simulink pentru platformele cu FPGA Zynq 7000

În urma instalării pachetelor de suport, mediul Matlab – Simulink va lansa în execuție formularul de configurare al platformei de dezvoltare cu procesor ARM și FPGA Zynq 7000. Se va alege opțiunea „ZedBoard”, după care se va continua procesul de configurare prin apăsarea butonului „Next” (Următoarea etapă). Formularul îndrumător, va configura generatoarele automate de cod, astfel încât toate bibleotecile și compilatoarele necesare vor fi incluse la fiecare generare de cod, sau la fiecare implementare a modelului Simulink pe platforma de dezvoltare. Toate setările se vor salva într-un fișier de configurare pe cardul de memorie SD.

Tot în cadrul acestor etape, se vor configura următorii parametrii:

- modul de acces al computerului gazdă la platforma de dezvoltare (prin rețea sau prin USB);
- instalarea sistemului de operare UNIX / LINUX pe cardul de memorie al platformei;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

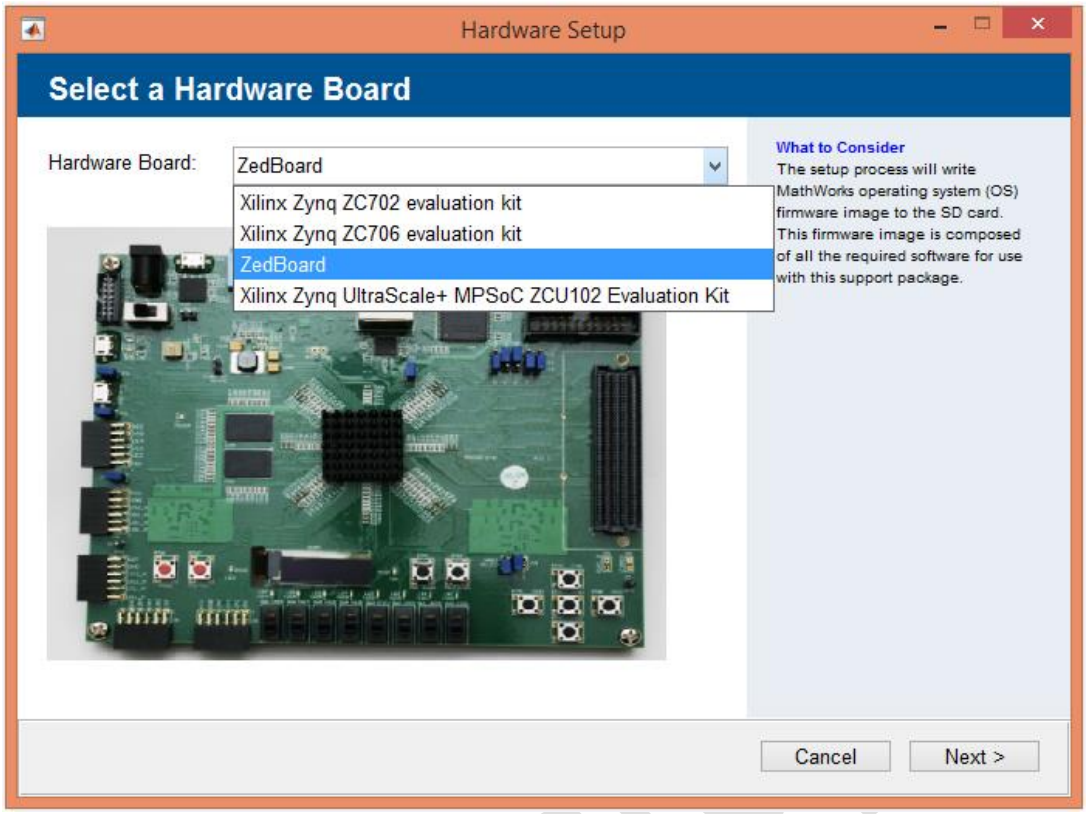


Fig. 54 – Configurarea platformei de dezvoltare ZedBoard Zynq 7000

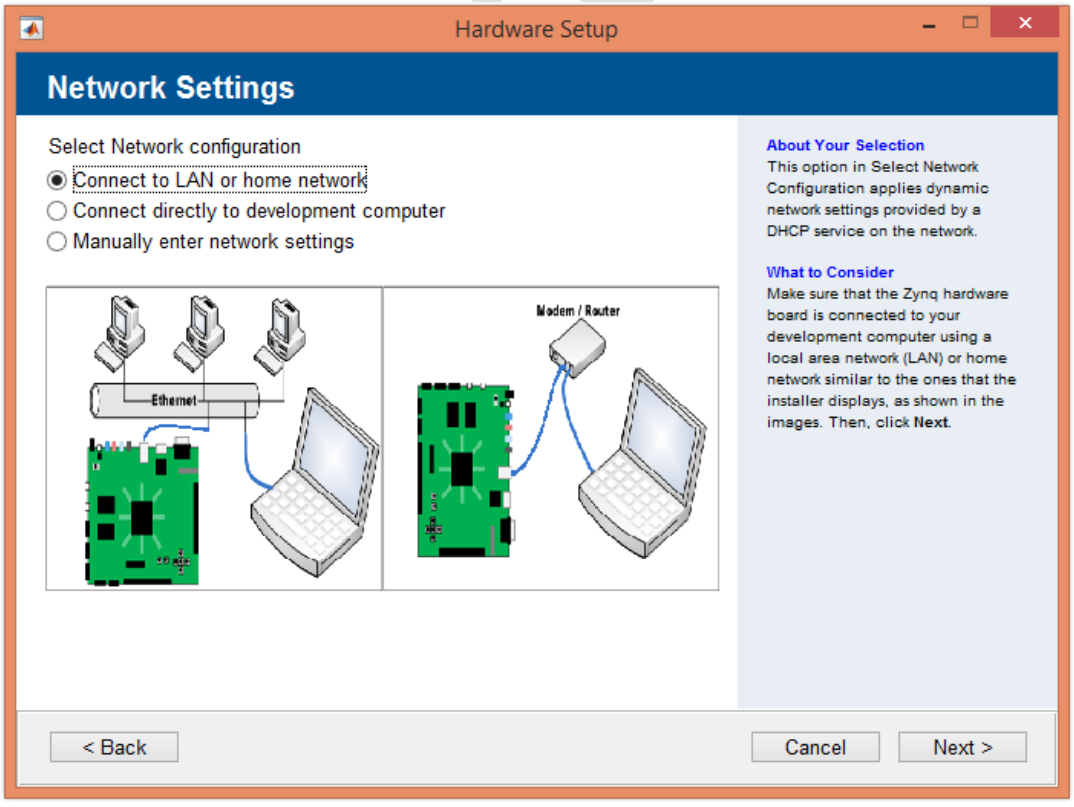


Fig. 55 – Configurarea modului de acces al computerului gazdă la platforma de dezvoltare

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

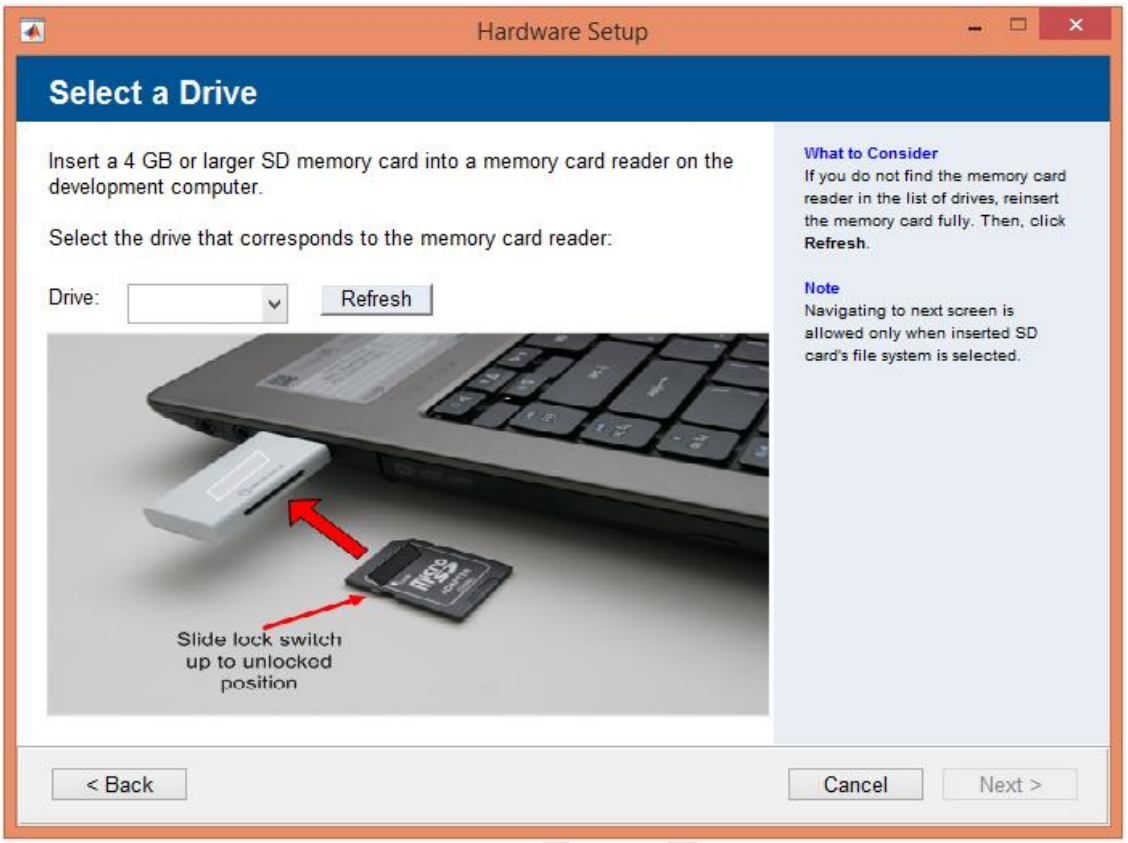


Fig. 56 – Instalarea sistemului de operare pe cardul de memorie SD

În urma instalării sistemului de operare pe cardul de memorie, se va putea proceda mai departe la implementarea unei aplicații de test pentru platforma ZedBoard. Pentru a realiza acest lucru, se va lansa în execuție mediul Matlab versiunea R2018b, și se vor introduce următoarele comenzi în consola de comandă Matlab:

```
hdlsetuptoolpath('ToolName','Xilinx
Vivado','ToolPath','C:\Xilinx\Vivado\2016.2\bin\vivado.bat');
```

- pentru a stabili mediul de sintetizare și compilare implicit Vivado 2016.2;

```
z = zynq;
```

- pentru a stabili conexiunea dintre calculatorul gazdă și platforma de dezvoltare;

```
z = z.setupZynqHardware
```

- pentru a configura parametrii de comunicare dintre platforma de dezvoltare și computer;

În urma comenzilor introduse, mediul Matlab va returna în cadrul consolei de comandă toate datele de identificare și de conectare la platforma de dezvoltare:

```
z =

  LinuxShell with properties:

    IPAddress: '76.45.136.111'
     Username: 'root'
         Port: 22
```

Astfel, prin intermediul datelor enumerate, utilizatorul va putea accesa consola de comandă (Terminalul) aferent sistemului de operare UNIX / Linux care rulează la nivelul platformei de dezvoltare. Acest lucru se poate realiza cu ajutorul unui program - client SSH, precum PuTTy.

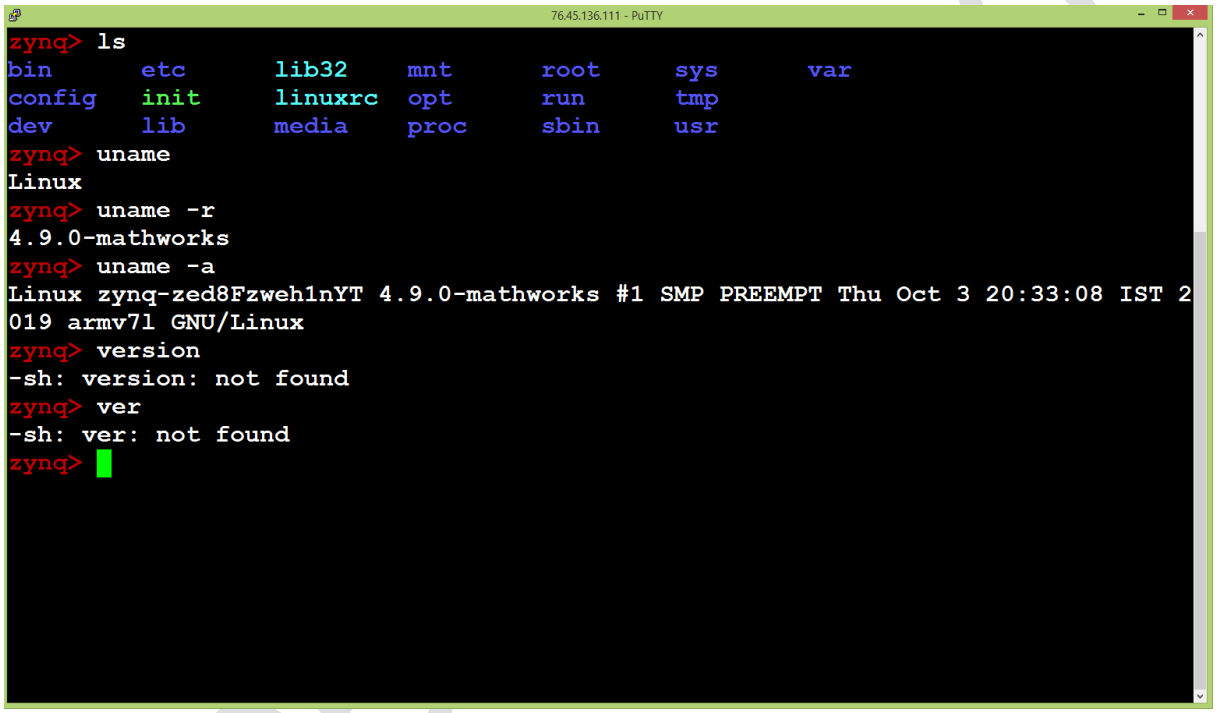


Fig. 57 – Consola de comandă a sistemului de operare UNIX / Linux de pe platforma de dezvoltare ZedBoard

Pentru a testa funcționalitatea platformei în cadrul mediului Matlab – Simulink, într-un director nou, gol al spațiului de lucru, se va introduce următoarea comandă în consola Matlab pentru a deschide o aplicație exemplu:

```
hdlcoder_led_blinking
```

În urma efectuării comenzii respective, se va lansa în execuție un model Matlab – Simulink, dezvoltat cu ajutorul elementelor din cadrul paletelor de instrumente:
- HDL Coder;
- Simulink;
- Embedded Coder;

Modelul dat, este un numărător reprezentat pe 8 biți. Logica de control a aplicației se regăsește în cadrul blocului „led_counter” (alabstru - verzui deschis). Conținutul blocului poate fi transpus în cod VHDL, pentru a programa aria de porți (eng. FPGA).

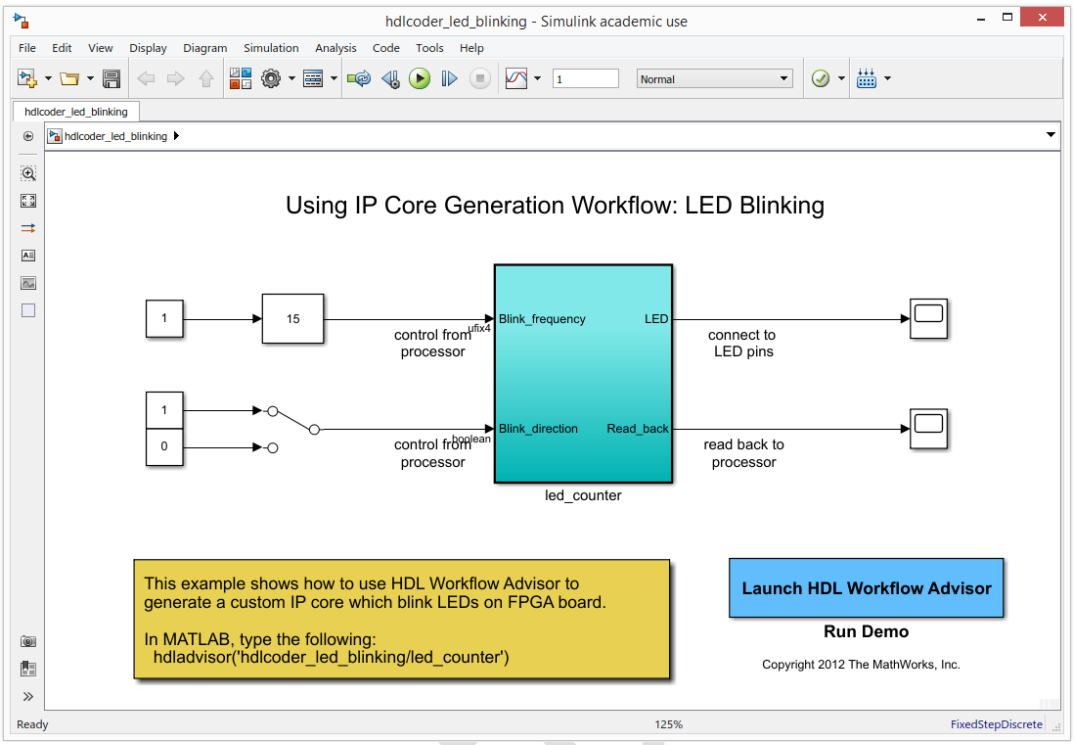


Fig. 58 – Modelul exemplu pentru programarea platformei ZedBoard

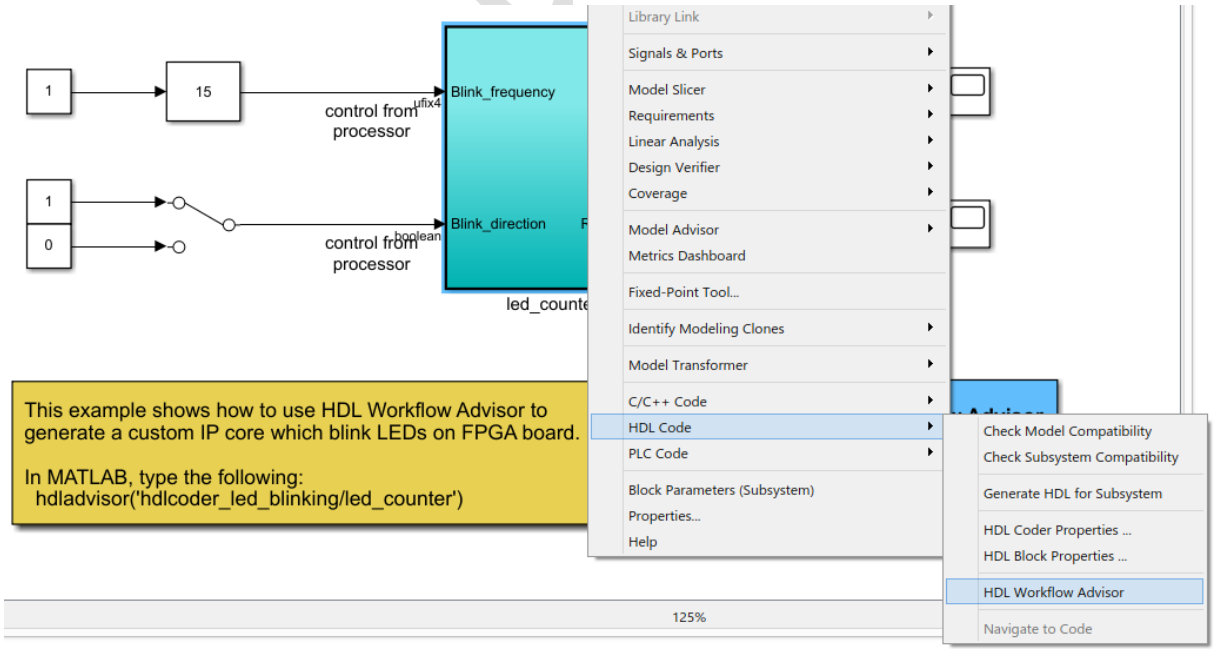


Fig. 59 – Apelarea îndrumătorului pentru generarea automată a codului VHDL

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Pentru a transpune în cod VHDL conținutul blocului „led_counter”, se va efectua operația „click – dreapta” asupra blocului, iar din cadrul meniului contextual „HDL Code” se va alege opțiunea „HDL Workflow Advisor”. Se va lansa în execuție îndrumătorul pentru configurare și generare automată de cod „HDL Workflow Advisor”.

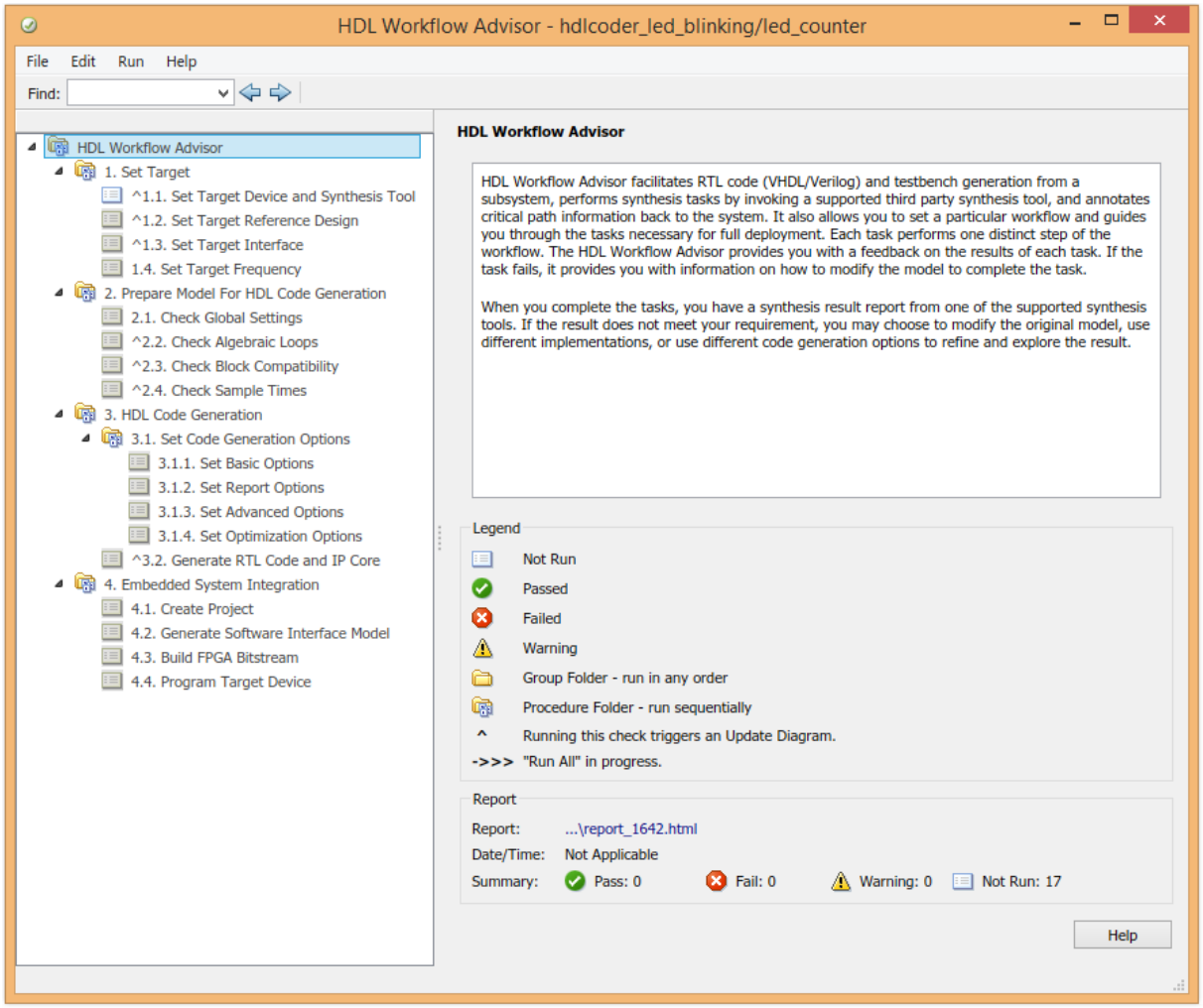


Fig. 60 – Fereastra îndrumătorului HDL Workflow Advisor

Etapele de execuție pentru procesului de generare automată a codului sunt structurate pe capitole principale precum:
- „Set Target” (configurarea platformei de dezvoltare);
- „Prepare Model for HDL Code Generation” (etapa pregătitoare pentru generarea de cod);
- „HDL Code Generation” (generarea efectivă a codului HDL);
- „Embedded System Integration” (implementarea aplicației la nivelul platformei);

Prin urmare, se vor parcurge etapele respective pentru a genera aplicația executabilă. În cadrul capitolului „Set Target”, în etapa „Set Target Device and Synthesis Tool” se vor selecta următoarele opțiuni:
- Target workflow: IP Core Generation (generare nucleu cu proprietate intelectuală dedicată);
- Target platform: ZedBoard;
- Synthesis tool: Xilinx Vivado;

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

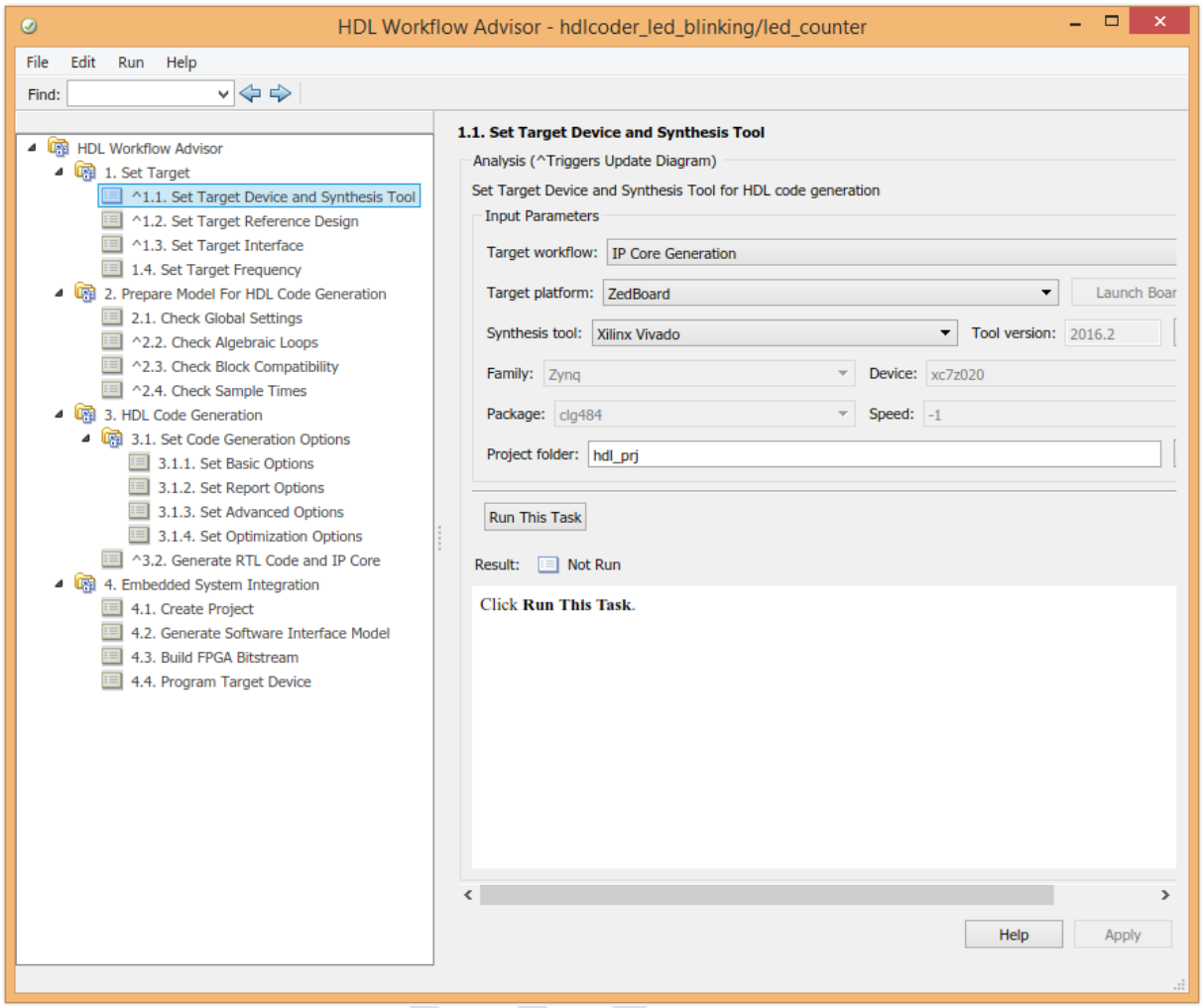


Fig. 61 – HDL Workflow Advisor - capitolul „Set Target”, etapa „Set Target Device and Synthesis Tool”

În cadrul etapei „Set Target Reference Design”, se vor păstra setările implicite, anume, „Reference design: Default system” (păstrarea valorilor implicite pentru proiectarea arhitecturii interne a platformei cu FPGA).

În cadrul etapei „Set Target Interface” ca și metodă de sincronizare se va alege opțiunea „Processor / FPGA synchronization: Free running”. Tot în cadrul acestei etape, se vor configura elementele periferice fizice (eng. hardware) utilizate în acest proiect, prin intermediul opțiunilor „Target platform interface table”. Se vor utiliza următoarele valori:

- „Blink_frequency: AXI4-Lite” (în categoria „Target Platform Interfaces”);
- „Blink_direction: AXI4-Lite” (în categoria „Target Platform Interfaces”);
- „LED: LEDs General Purpose [0:7] (în categoria „Target Platform Interfaces”);
- „Read_back: AXI4-Lite” (în categoria „Target Platform Interfaces”);

În categoria „Set Target Frequency” se vor păstra valorile implicite, anume „50 [MHz]. Pentru a aplica toate modificările din cadrul acestui capitol, se va efectua operația „click – dreapta” asupra titlului capitolului, anume „Set Target” iar din meniul contextual, se va alege opțiunea „Run All”.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

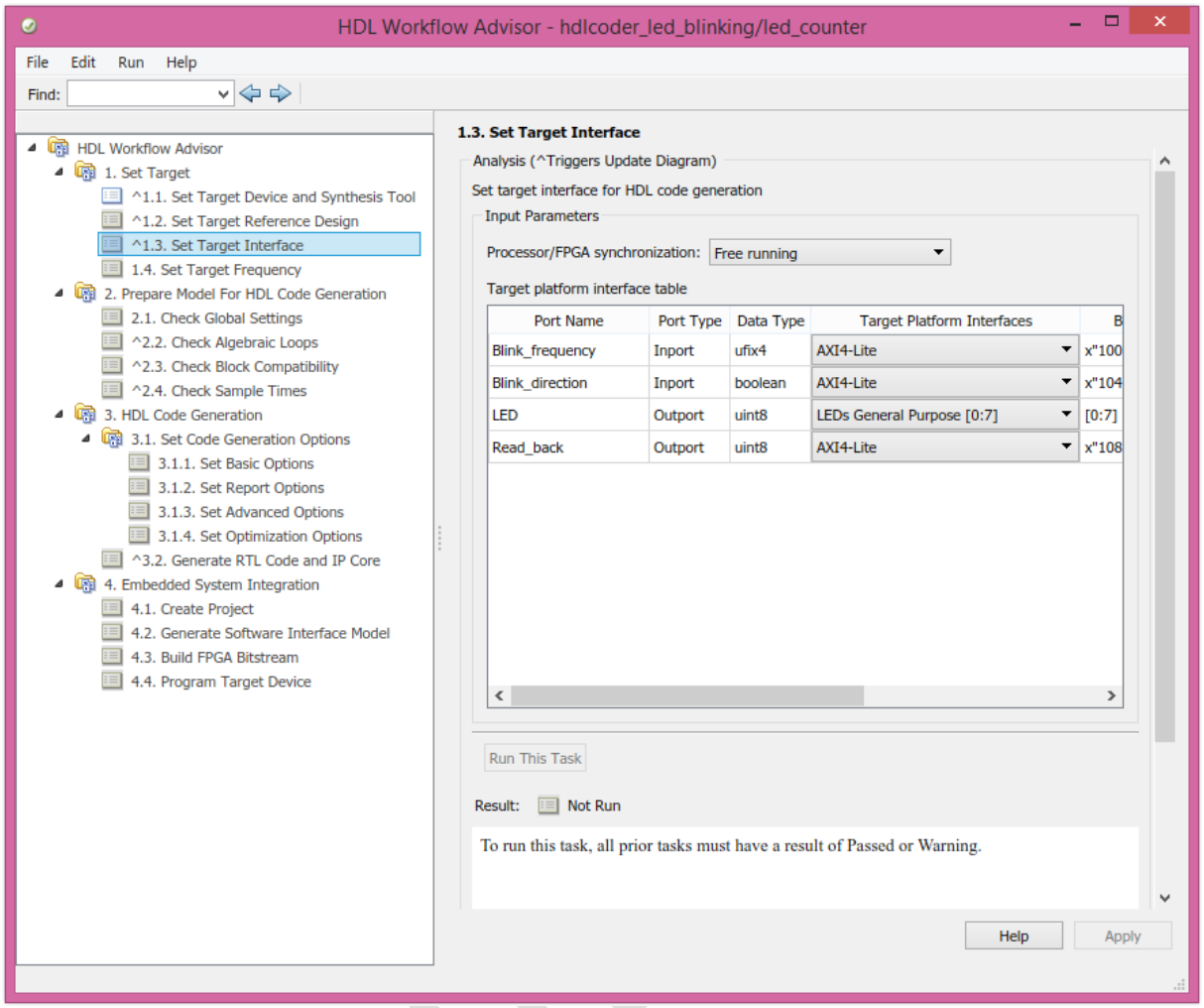


Fig. 62 – HDL Workflow Advisor - capitolul „Set Target”, etapa „Set Target Interfaces”

Pentru capitolul „Prepare Model for HDL Code Generation” se vor păstra valorile implicite recomandate, și se va parcurge etapa de verificare și executare prin selectarea opțiunii „Run All” din cadrul meniului contextual obținut prin operația „click – dreapta”.

Se va proceda în mod similar pentru etapa „HDL Code Generation”. La finalizarea execuției etapei respective, se va obține o fereastră de raport care va descrie tot procesul care a avut loc în cadrul etapei pentru generare a codului – program.

Ultimele etape, din cadrul ultimului capitol, anume „Embedded System Integration”, se vor parcurge fiecare în parte, pe rând, prin selectarea opțiunii „Run This Task” regăsită sub formă de buton în cadrul etapei respective. Se vor parcurge astfel etapele:

- „Create Project” (creeare proiect Vivado);
- „Generate Software Interface Model” (generare model Simulink pentru calculatorul gazdă);
- „Build FPGA Bitstream” (creearea fișierului binar pentru configurarea ariei de porți);
- „Program Target Device” (programarea sau implementarea fișierului binar – executabil);

În etapa de generare a fișierului binar - executabil pentru configurarea ariei de porți, va fi invocat mediul de programare HDL Vivado 2016.2 la nivel de consolă de comandă. Prin intermediul mediului Vivado 2016.2, se va parcurge procesul de sintetizare, generare și implementare a codului HDL generat.

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

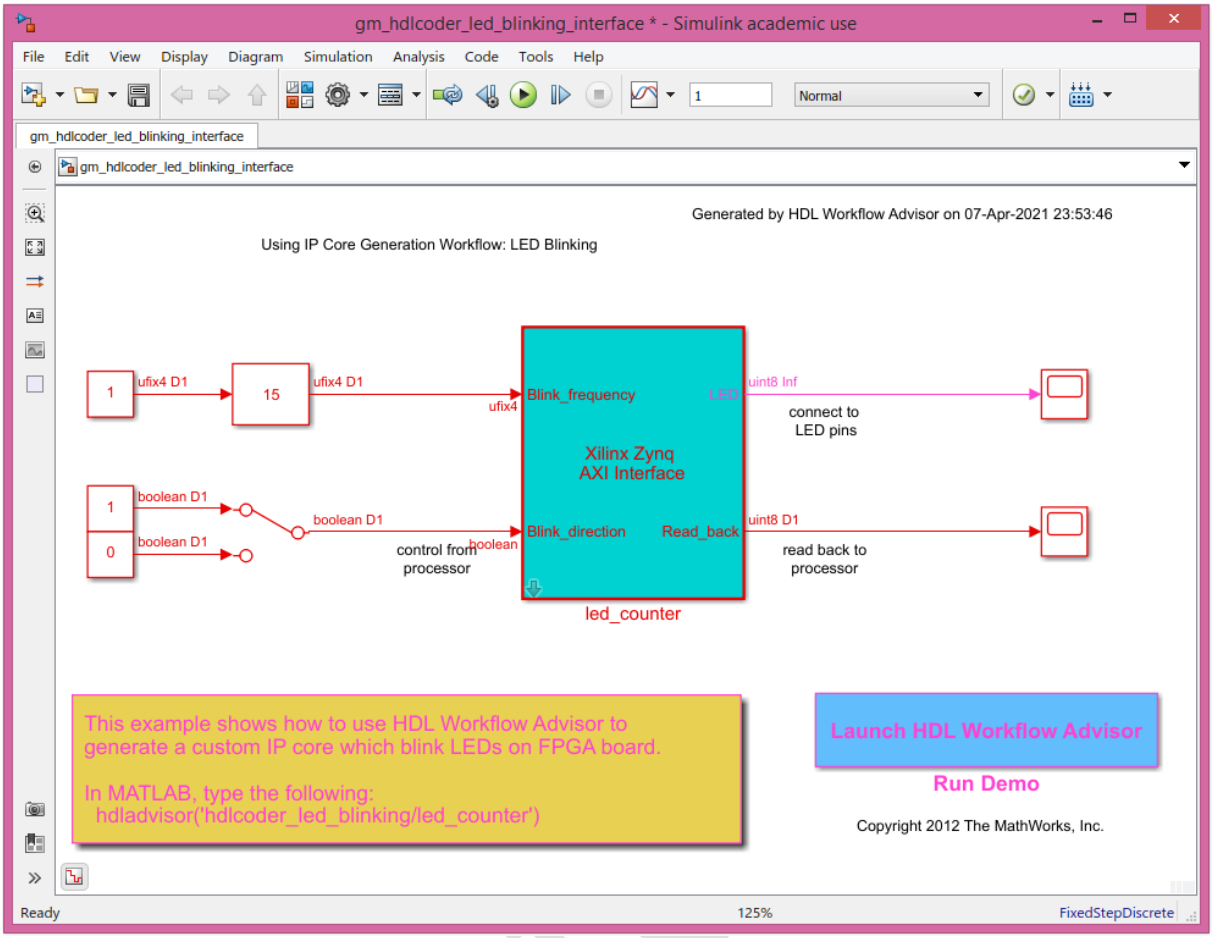


Fig. 63 - Modelul Simulink pentru calculatorul gazdă pentru controlul aplicației în timp real

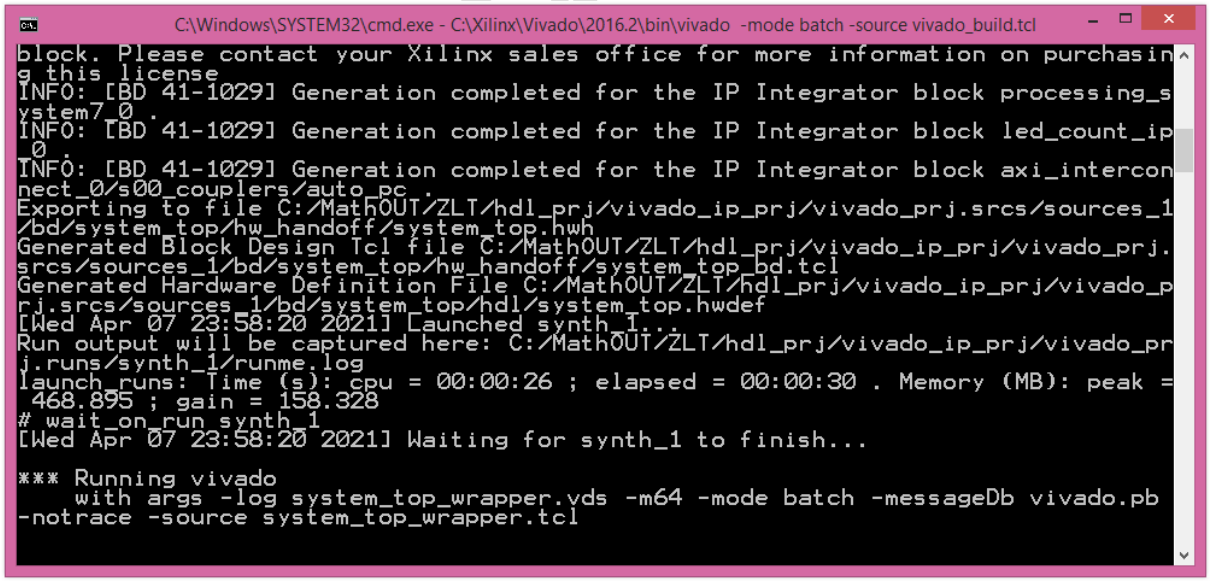


Fig. 64 – Invocarea mediului Vivado 2016.2 la nivel de consolă în vederea generării fișierului binar - executabile pentru configurarea ariei de porți

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

Operația post – sinteză de programare efectivă (de transferare în memoria platformei a fișierului binar - executabil), se va realiza prin intermediul ultimei etape „Program Target Device”. Se va selecta operația, după care, se va parcurge prin opțiunea „Run This Task”. În urma efectuării tuturor etapelor, îndrumătorul HDL Workflow Advisor va bifa toate etapele.

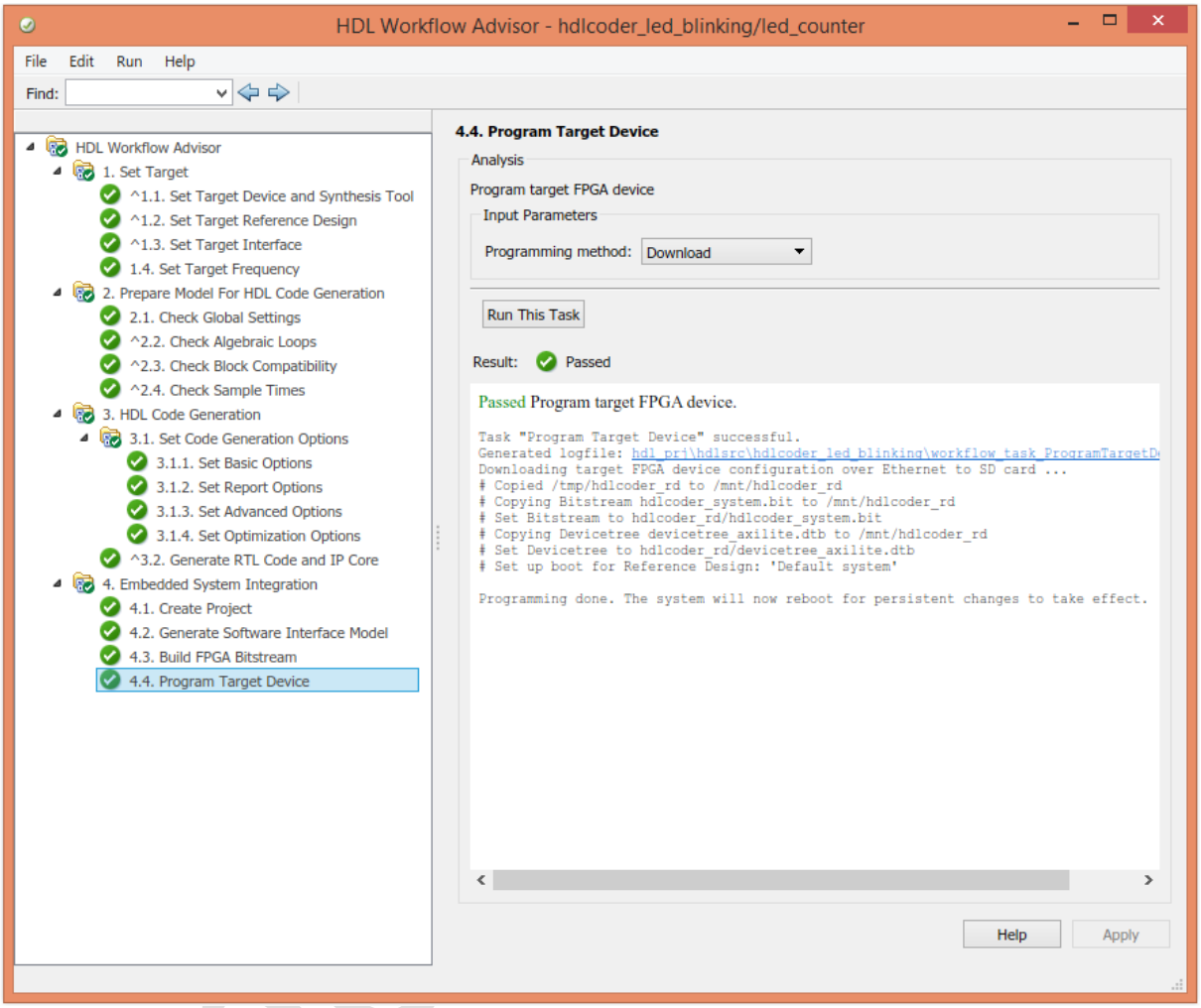


Fig. 65 – Parcurgerea completă a tuturor etapelor din cadrul îndrumătorului HDL Workflow Advisor

Pentru a inițializa aplicația la nivel de platformă de dezvoltare, este necesară, programarea și a procesorului de aplicație ARM. Acest lucru, se poate realiza, prin intermediul modelului Simulink pentru calcualtorul gazdă, generat anterior. În cadrul modelului respectiv, se va alege timpul total de simulare „inf” (infinit), iar modul de execuție „External” (extern – executare la nivel de platformă de dezvoltare).

În urma efectuării parametrizărilor indicate asupra modelului de control, din cadrul paletei de instrumente se va alege opțiunea „Deploy to Hardware” (generarea și încărcarea codului program la nivelul procesorului de aplicație ARM al platformei ZedBoard).

Pentru a interacționa în timp real cu parametrii programului executat la nivel de platformă de dezvltare se va alege opțiunea „Run” (butonul PLAY verde din barea de instrumente Simulink).

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

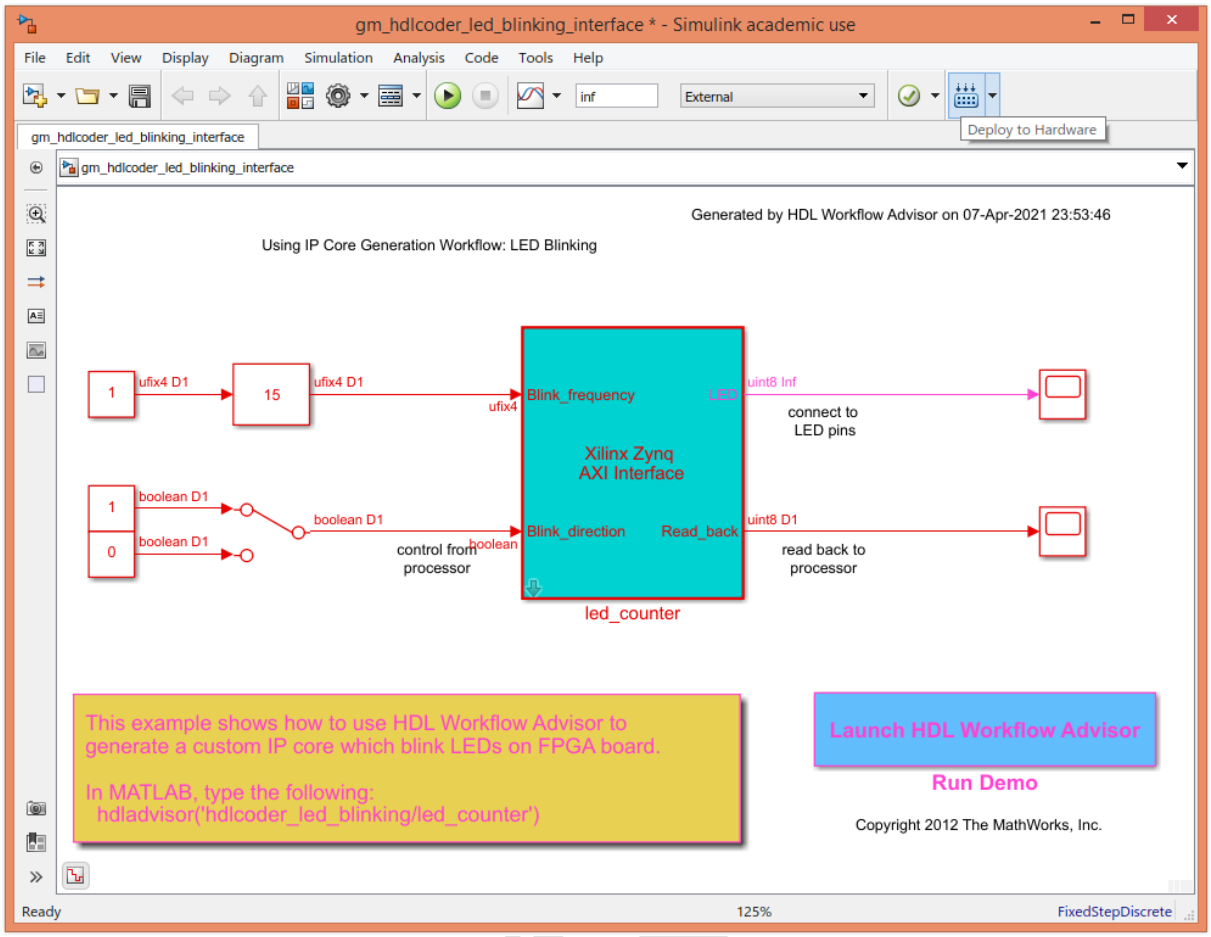


Fig. 66 – Parametrizarea modelului de control aferent calculatorului gazdă

Prin intermediul cursorului „Slider Gain” se poate adjusta frecvența numărătorului, prin intermediul comutatorului, sensul de numărare, iar prin intermediul osciloscopului se poate urmări rezultatul obținut în urma executării codului – program.

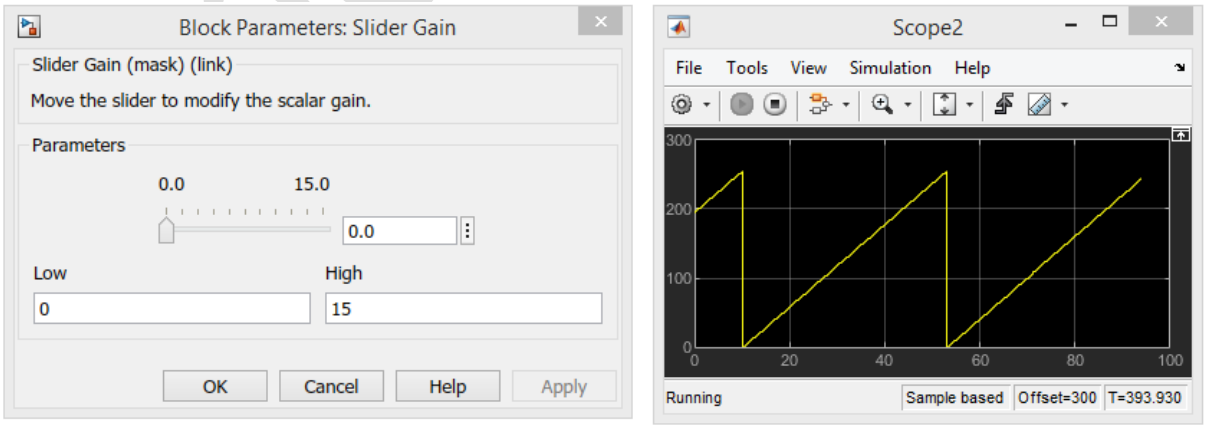


Fig. 67 – Elementele de interacțiune în timp real cu parametrii aplicației ce rulează pe platforma ZedBoard – Zynq 7000

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP”
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

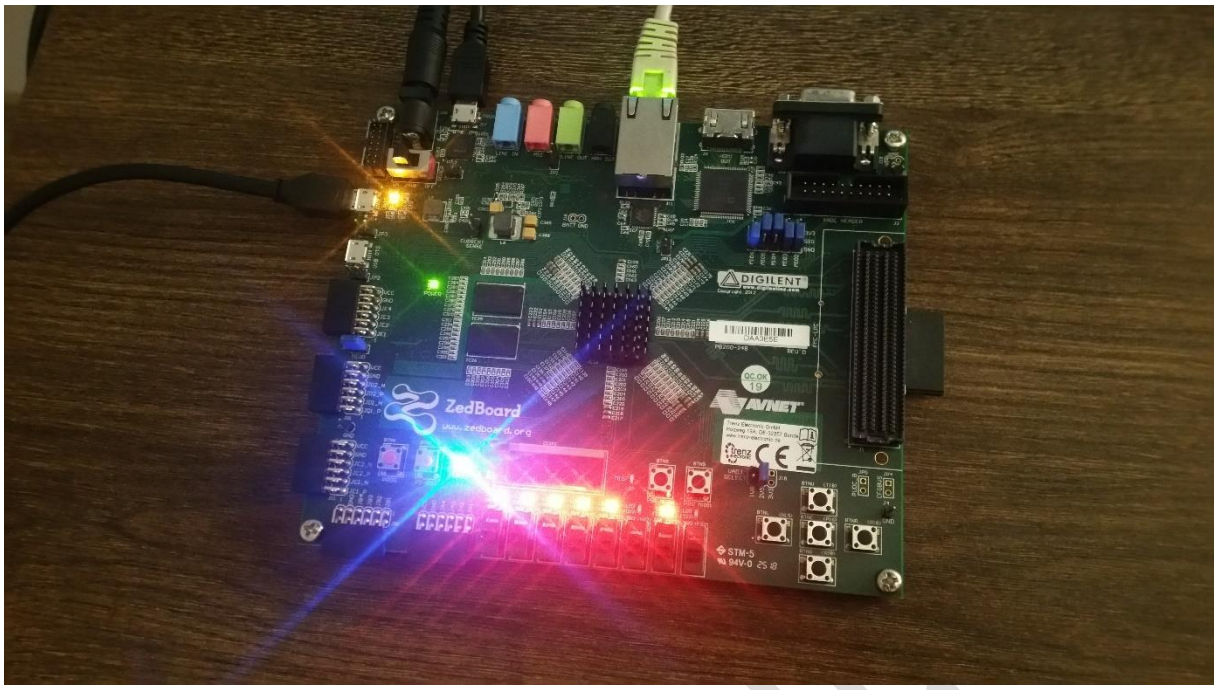


Fig. 68 – Aplicația rulând la nivel de platformă de dezvoltare

IV. CONCLUZIE:

Spre deosebire de celelalte sisteme de calcul (micro-controller, DSP sau microprocesor), un sistem de calcul pe bază de FPGA, în urma procesului de „programare" **își modifică arhitectura fizică** (hardware) **internă** astfel încât, codul – program să poată fi implementat la nivelul capsulei FPGA sub forma unui circuit logic ultra rapid.

V. BIBLIOGRAFIE:

1. Teodor Crișan Pană – „Sisteme de calcul cu microprocesoare, FPGA și DSP" – Editura UTPRESS, Cluj – Napoca, 2016 – ISBN 978-606-737-206-9;

2. Ioana – Cornelia GROS, Lucian – Nicolae PINTILIE, Teodor Crișan PANĂ – „SISTEME EMBEDDED ÎN INGINERIE ELECTRICĂ - GHID DE APLICAȚII" – Editura UTPress Cluj – Napoca, 2020 ISBN 978-606-737-431-5:
(https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/431-5.pdf);

3. CircuitDigest – FPGA Structure:
(https://circuitdigest.com/sites/default/files/inlineimages/u1/FPGA-Structure.png);

4. YouTube – „How to Get Started with Basys 3 Board and Vivado| FPGA for BEGINNERS":
(https://www.youtube.com/watch?v=tOwMmBI_XNo);

5. Austin H. Duncan – East Tennessee State University - Undergraduate Honors Theses - Student Works - Digital Commons – „Logic Gates Using the Digilent Basys3";

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro

(https://dc.etsu.edu/cgi/viewcontent.cgi?article=1327&context=honors);

6. Reichlet - ZedBoard Zynq-7000 ARM/FPGA SoC development board: (https://www.reichelt.com/de/en/zedboard-zynq-7000-arm-fpga-soc-development-board-digil-410-248-p243344.html);

7. MathWorks.com – „Run a Simulink Model on Zynq – series" – Videos and Webinars: (https://www.mathworks.com/videos/run-a-simulink-model-on-zynq-introduction-and-requirements-1-of-4-89508.html);

8. MathWorks.com – „Xilinx Zynq Support from MATLAB and Simulink" – Hardware Support: (https://www.mathworks.com/hardware-support/zynq.html);

Realizat de: ing. drd. Pintilie Lucian - Nicolae
Pentru disciplina: „Sisteme cu FPGA și DSP"
Adresă de e-mail: Lucian.Pintilie@emd.utcluj.ro