****

**Controller de tastatura PS2**

**Universitatea Tehnica Cluj-Napoca**

**Facultatea de Automatica si Calculatoare**

**Sectia Calculatoare, Seria B, grupa 30219**

**Proiect realizat de Marchis Raul si Martin Darius**

**Profesor coordonator : Laurentiu Opincariu**

**Cuprins**

**1.Specificatiile proiectului**

**2.Schema Block**

**3.Scan codes**

**4.Componentele utilizate**

**5.Lista intrari/iesiri**

**6.Justificarea solutiei alese**

**7.Instructiuni de utilizare**

**8.Posibilitati de dezvoltare ulterioara**

**9.Bibliografie**

**1.Specificatiile proiectului**

**Cerinta proiectului** : Sa se proiecteze un **controller de tastatura PS2.** Se cere citirea tastelor si afisarea caracterelor corespunzatoare pe afisajul cu 7 segmente. Se vor vedea ultimele 4 simboluri, iar tastele de control vor avea roluri speciale (ex. : Enter-ul incepe un rand nou, stergand afisajul).

Documentatie: manualele de referinta pentru placile cu FPGA si documentatia pentru protocolul PS2.

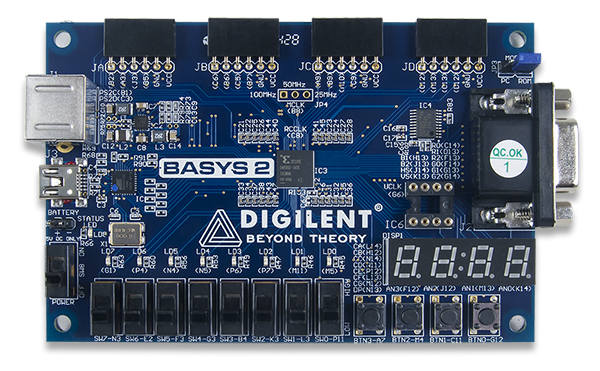
<http://www.computer_engineering.org/ps2protocol/>. Proiectul va fi realizat **de 2 studenti.**

1. Shiftarea datelor pe parcurgerea introducerilor (Ex : STINS STINS STINS STINS ; push A => STINS STINS STINS A; push B => STINS STINS A B.
2. Enter-ul = caracter special, apasarea lui duce la resetarea afisorului.

**Obiectele hardware folosite pentru proiect :** Tastatura PS/2 cu 6 pini , placuta FPGA Nexys 2 / Basis 2, PC , mouse.

Pentru dezvoltarea acestui proiect s-au folosit: mediul de dezvoltare Xilinx ISE 13.4 și limbajul de descriere hardware VHDL.

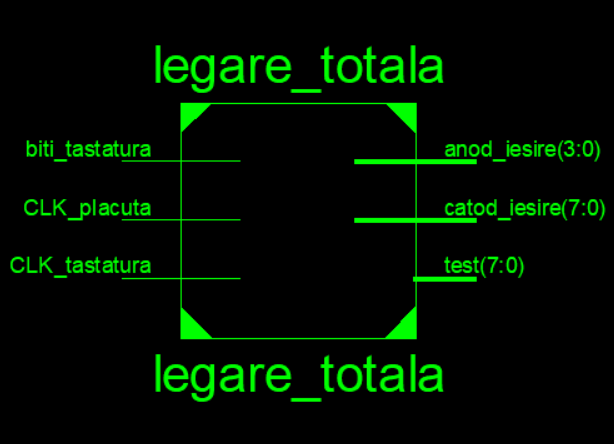


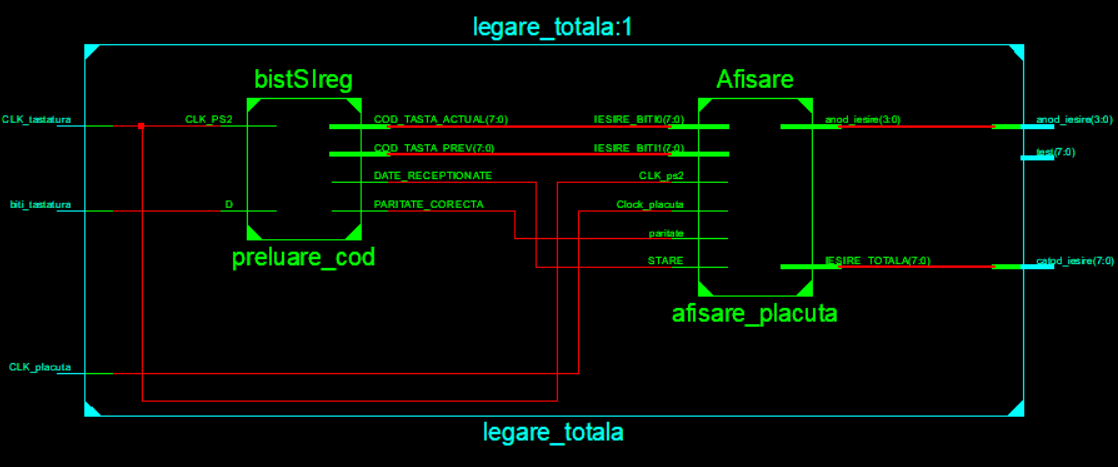


Domeniul de studiu este unul rezervat lucrului exclusiv cu componente periferice și sisteme de I/O(Input/Output). Protocolul folosit, în cazul nostru, tastatura, înțelege caracterele date sub forma unui cod în binar transmis de la periferic. Conectarea se poate face prin diferite tipuri de porturi, unul din ele fiind PS/2(Personal System 2).

Ceea ce dorim noi defapt sa facem este sa “programam” receptia corecta a datelor primite la apasarea unei taste, lucru pentru care e nevoie de placuta DIGILENT Basys2 / Nexys2.

**2.Schema Block**





Când ambele linii de comunicații sunt high conexiunea se află în starea „inactiv. Clock-ul placutei trebuie sa fie ‘1’ logic, iar clock-ul tastaturii ‚0’ (în această situație tastatura are permisiunea de a începe transmiterea datelor) pentru a incepe receptia bit cu bit a secventei „make-code” transmisa. Semnalul de tact al tastaturii este cuprins între 10 și 16.7 kHz și este esențial ca unitatea de control să țină cont de această frecventă în transmiterea/recepția datelor. Una dintre cele mai importante detalii o reprezinta sincronizarea clock-ului tastaturii pentru receptia datelor, impreuna cu o buna cunoastere a modului de transmisie.

Un pachet trimis de către tastatură catre dispozitivul de control conține 11 biți:

1. Bit-ul de start : ‘0’
2. Byte-ul ce contine „make-code”-ul tastei apasate : ex: pentru tasta 0 => 01000101(45). Totusi, tastatura trimite inspre host aceasta combinatie de biti „invers” : in loc de 45, vom primi pe rand secventa „01010100”, adica 54. Astfel, ordinea bitilor va fi luata in considerare de registrul universal.
3. Un bit de paritate(poate fi 0 sau 1 in functie de numarul de biti de 1 din „make-code” : trebuie sa avem un numar impar de biti de 1 in acel byte ca bit-ul de paritate sa fie 0 => paritate impara).
4. Bit-ul de stop : ‚1’.

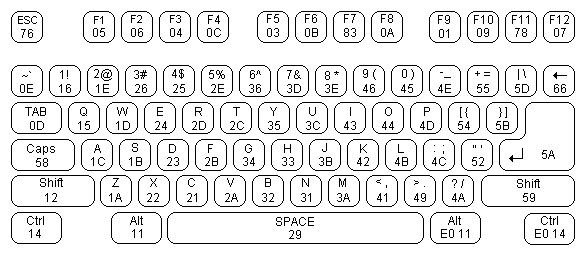
**Exemplu de cod transmis :** Push 0 => make-code = „0 01010100 0 1”.

Biții start si stop sunt folositi pentru a semnaliza inceputul / sfarsitul receptiei datelor. Totodata, pentru a putea trece la afisarea tastelor pe placuta, avem nevoie de certitudinea ca receptia bitilor a avut loc (ca au fost transmiti toti cei 11 biti), cat si ca acestia au fost transmisi corect. O buna modalitate de verificare a acestui lucru poate fi facuta astfel : retinem intr-un semnal operatia “sau-exclusiv” intre toti cei 8 biti care formeaza “make-code”-ul. Daca rezultatul este 1, inseamna ca a avut loc transmisia corecta.

Transmisia datelor spre host se face in acest mod :   
 PUSH 0 => 45 (ce reprezinta “make-code”-ul), urmand de F045 (ce reprezinta “break code”-ul). Aceast lucru se genereaza in momentul in care dam drumul butonului.

**3.Scan codes**

Pentru implementarea modulelor care compun controller-ul interfetei PS/2 pentru tastatura s-a folosit mediul de dezvoltare Xilinx ISE produs de compania Xilinx. Ca limbaj de proiectare a arhitecturii hardware s-a folosit unul dintre cele mai complete limbaje de descriere hardware : VHDL.

**“MAKE-CODE”-ul tastelor**

**4.COMPONENTELE UTILIZATE**

Registru pentru receptia datelor (BistSiReg)

Divizor de frecventa (pentru afisare)

Afisarea pe decodificatorul BCD 7-segmente

**BistSiReg**

Componenta receptioneaza transmiterea datelor de la tastatura inspre host.  
Aceasta stabileste si receptia totala a datelor, cat si corectitudinea transmisiei acestora. In plus, este realizata shiftarea afisoarelor intre ele.

Semnalele entitatii bistSIreg sunt :  
 CLK\_PS2 : in std\_logic;  
 D : in std\_logic;  
 COD\_TASTA\_ACTUAL : out bit\_vector(7 downto 0);

COD\_TASTA\_PREV : out bit\_vector(7 downto 0);

DATE\_RECEPTIONATE: out std\_logic;

PARITATE\_CORECTA: out bit);

Semnalele externe :  
 Bistabil\_trecere : std\_logic\_vector (1 downto 0);  
 RECEPTIE\_BITI\_DATE: std\_logic\_vector (10 downto 0);  
 bit\_paritate: bit;

Semnalul CLK\_PS2 este semnalul de clock al tastaturii, prin D receptionam informatia bit cu bit de la tastatura. COD\_TASTA\_ACTUAL il folosim pentru a scoate informatia Make-code-ului din aceasta component, iar COD\_TASTA\_PREV pentru a scoate F0-ul. DATE\_RECEPTIONATE il folosim pentru a ne asigura ca am primit toti cei 11 biti, iar PARITATE\_CORECTA pentru a ne asigura ca informatia este corecta.

**COD :**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

use IEEE.NUMERIC\_STD.all;

entity bistSIreg is

port(

CLK\_PS2 : in std\_logic;

D : in bit;

COD\_TASTA\_ACTUAL : out bit\_vector(7 downto 0);

COD\_TASTA\_PREV : out bit\_vector(7 downto 0);

DATE\_RECEPTIONATE: out std\_logic;

PARITATE\_CORECTA: out bit);

end;

architecture arch\_bist of bistSIreg is

signal Bistabil\_trecere : std\_logic\_vector ( 1 downto 0);

signal RECEPTIE\_BITI\_DATE: std\_logic\_vector(10 downto 0);

signal bit\_paritate:bit;

begin

-- process(CLK)

-- begin

-- if (CLK = '1' and CLK'EVENT) then

-- Bistabil\_trecere(0) <= CLK\_PS2;

-- Bistabil\_trecere(1) <=D;

-- end if;

-- end process;

process(CLK\_PS2, D)

variable count : std\_logic\_vector(3 downto 0) := "0000";

variable RECEPTIE\_BITI : bit\_vector(10 downto 0) := "11111111111";

variable temp\_prev : bit\_vector(7 downto 0) := "11111111";

variable temp\_actual : bit\_vector(7 downto 0) := "11111111";

begin

if (CLK\_PS2'event and CLK\_PS2 = '0') then

count := count + "0001";

RECEPTIE\_BITI := RECEPTIE\_BITI sll 1;

RECEPTIE\_BITI(0) := D;

DATE\_RECEPTIONATE <= '0';

if (count = "1011") then

temp\_prev := temp\_actual; -- temp\_prev este F0, adica break code

temp\_actual(0) :=RECEPTIE\_BITI(9);

temp\_actual(1) :=RECEPTIE\_BITI(8);

temp\_actual(2) :=RECEPTIE\_BITI(7);

temp\_actual(3) :=RECEPTIE\_BITI(6);

temp\_actual(4) :=RECEPTIE\_BITI(5);

temp\_actual(5) :=RECEPTIE\_BITI(4);

temp\_actual(6) :=RECEPTIE\_BITI(3);

temp\_actual(7) :=RECEPTIE\_BITI(2);

PARITATE\_CORECTA <= RECEPTIE\_BITI(9) xor RECEPTIE\_BITI(8) xor RECEPTIE\_BITI(7) xor RECEPTIE\_BITI(6) xor RECEPTIE\_BITI(5) xor RECEPTIE\_BITI(4) xor RECEPTIE\_BITI(3) xor RECEPTIE\_BITI(2) xor RECEPTIE\_BITI(1);

DATE\_RECEPTIONATE <= '1';

RECEPTIE\_BITI := "11111111111";

count:="0000";

end if;

end if;

COD\_TASTA\_PREV <= temp\_prev; -- F0

COD\_TASTA\_ACTUAL <= temp\_actual; --Make - code

end process;

end arch\_bist;

**DIVIZORUL DE FRECVENTA** Realizat cu scopul de a asigura afisajul corect al tastelor pe placuta : clock-ul placutei Basys2 este de 100 MHz, iar clock-ul tastaturii intre 10 – 16.7 KHz. Rezulta astfel necesitatea de a “diviza” (imparti) clock-ul placutei in “range-ul” tastaturii : 216. Consideram doar cei mai semnificativi doi biti ai “numaratorului” pentru a asigura incetinirea semnalului de tact al tastaturii.

COD

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity Divizor\_SEG is

port(Clock: in std\_logic;

Clock\_Divizat: out std\_logic\_vector(1 downto 0));

end Divizor\_SEG;

architecture Arch\_Div\_SEG of Divizor\_SEG is

signal Num: std\_logic\_vector(16 downto 0);

begin

process(Clock)

begin

if Clock ='1' and Clock'event then

Num<=Num+1;

end if;

end process;

Clock\_Divizat<=Num(16 downto 15);

End Arch\_Div\_SEG;

**Afisarea**

Componenta “Afisare” se utilizeaza pentru a sincroniza afisoarele, cat

si pentru a “seta” pe catozi cod-ul de 8 biti (cele 7 liniute cat si punctul).

In aceasta componenta scan code-urile se transforma in catodul care trebuie afisat.

Semnalele interne ale entitatii:   
 Clock\_placuta: in std\_logic;

STARE, PARITATE: in std\_logic;

IESIRE\_BITI0: in bit\_vector(7 downto 0);

IESIRE\_BITI1: in bit\_vector(7 downto 0);

IESIRE\_TOTALA: out std\_logic\_vector(7 downto 0);

anod\_iesire: out std\_logic\_vector(3 downto 0);

CLK\_ps2: in std\_logic);

Toate semnalele de intrare le primim din component BistSIReg, IESIRE\_TOTALA reprezinta catodul (cele 7 segmente+ punctual zecimal), anod\_iesire reprezinta anozii pe care se afiseaza informatia.

**Codul**:

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity Afisare is

port( Clock\_placuta: in std\_logic;

STARE, PARITATE: in std\_logic;

IESIRE\_BITI0: in bit\_vector(7 downto 0); -- apelam cu make code

IESIRE\_BITI1: in bit\_vector(7 downto 0); -- apelam cu break code

IESIRE\_TOTALA: out std\_logic\_vector(7 downto 0);

anod\_iesire: out std\_logic\_vector(3 downto 0);

CLK\_ps2: in std\_logic);

end entity;

architecture ARH\_SEG of Afisare is

signal bcd1, bcd2, bcd3, bcd4 : std\_logic\_vector(7 downto 0);

component Divizor\_SEG

port(Clock: in std\_logic;

Clock\_Divizat: out std\_logic\_vector(1 downto 0));

end component;

signal Clock\_Divizat : std\_logic\_vector (1 downto 0);

begin

div\_frecventa : Divizor\_SEG port map (Clock\_placuta, Clock\_Divizat);

alt\_proc: process(CLK\_ps2)

variable cod1 : std\_logic\_vector (7 downto 0) := "11111111";

variable cod2 : std\_logic\_vector (7 downto 0) := "11111111";

variable cod3 : std\_logic\_vector (7 downto 0) := "11111111";

variable cod\_trebuinta : std\_logic\_vector (7 downto 0) := "11111111";

begin

if (stare='1' and paritate='1' and IESIRE\_BITI1="11110000" and CLK\_PS2'event and CLK\_PS2='1' ) then -- ultimele 2 verificari sunt pentru ca, dupa ce se genereaza F01C de exemplu, clock-ul tastaturii ramane la 1 mereu pana urmeaza o alta tasta

cod3 := cod2;

cod2 := cod1;

cod1 := cod\_trebuinta; -- cod\_trebuinta inca e VECHEA VALOAREA, dupa case o sa primeasca noua tasta apasata

case IESIRE\_BITI0 is

when "01000101" => cod\_trebuinta := "10000001"; -- 0

when "00010110" => cod\_trebuinta := "11001111"; -- 1

when "00011110" => cod\_trebuinta := "10010010"; -- 2

when "00100110" => cod\_trebuinta := "10000110"; -- 3

when "00100101" => cod\_trebuinta := "11001100"; -- 4

when "00101110" => cod\_trebuinta := "10100100"; -- 5

when "00110110" => cod\_trebuinta := "10100000"; -- 6

when "00111101" => cod\_trebuinta := "10001111"; -- 7

when "00111110" => cod\_trebuinta := "10000000"; -- 8

when "01000110" => cod\_trebuinta := "10000100"; -- 9

when "00100100" => cod\_trebuinta := "10110000"; -- E

when "00101011" => cod\_trebuinta := "10111000"; -- F

when "00111100" => cod\_trebuinta := "11000001"; -- U

when "01000011" => cod\_trebuinta := "11001111"; -- I

when "01000100" => cod\_trebuinta := "10000001"; -- O

when "01001101" => cod\_trebuinta := "10011000"; -- P

when "00110011" => cod\_trebuinta := "11001000"; -- H

when "01001011" => cod\_trebuinta := "11110001"; -- L

when "00011100" => cod\_trebuinta := "10001000"; -- A

when "00110101" => cod\_trebuinta := "11000100"; -- Y

when "01001001" => cod\_trebuinta := "01111111"; --.

when "01011010" => cod\_trebuinta := "11111111"; -- ENTER KEY

cod3 := "11111111";

cod2 := "11111111";

cod1 := "11111111";

when others => cod\_trebuinta := "10110000"; --doar acestea le putem afisa => restul le punem la 00000000

end case;

end if;

bcd4 <= cod3;

bcd3 <= cod2;

bcd2 <= cod1;

bcd1 <= cod\_trebuinta;

end process;

PROCES: process(Clock\_Divizat, bcd4, bcd3, bcd2, bcd1)

begin

case Clock\_Divizat is

when "00" => anod\_iesire(0)<='0' ; anod\_iesire(1)<='1' ; anod\_iesire(2)<='1'; anod\_iesire(3)<='1'; IESIRE\_TOTALA <= bcd4;

when "01" => anod\_iesire(0)<='1' ; anod\_iesire(1)<='0' ; anod\_iesire(2)<='1'; anod\_iesire(3)<='1'; IESIRE\_TOTALA <= bcd3;

when "10" => anod\_iesire(0)<='1' ; anod\_iesire(1)<='1' ; anod\_iesire(2)<='0'; anod\_iesire(3)<='1'; IESIRE\_TOTALA <= bcd2;

when "11" => anod\_iesire(0)<='1' ; anod\_iesire(1)<='1' ; anod\_iesire(2)<='1'; anod\_iesire(3)<='0'; IESIRE\_TOTALA <= bcd1;

when others => anod\_iesire(0)<='1' ; anod\_iesire(1)<='1' ; anod\_iesire(2)<='1'; anod\_iesire(3)<='1'; -- Nu conteaza ..

end case;

end process;

end ARH\_SEG;

**Legarea componentelor : componenta Unity** Folosita pentru a transmite si modifica semnalele principale:   
 Semnalele interne ale entitatii:  
 biti\_tastatura : in std\_logic;  
 CLK\_tastatura : in std\_logic;  
 CLK\_placuta : in std\_logic;  
 catod\_iesire : out std\_logic\_vector (7 downto 0);  
 anod\_iesire : out std\_logic\_vector (7 downto 0);

Semnalele externe :   
 signal temp\_break, temp\_make : bit\_vector (7 downto 0) := "11111111";

signal paritate : std\_logic;

signal receptie\_date : std\_logic;

Intrarile in aceasta unitate sunt Clock-urile de la placuta si de la tastatura si intrarea de date de la tastatura, iar iesirile sunt catodul si anozii placutei.

**Codul:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity legare\_totala is

port(biti\_tastatura : in bit;

CLK\_tastatura : in std\_logic;

CLK\_placuta : in std\_logic;

catod\_iesire : out std\_logic\_vector(7 downto 0);

anod\_iesire : out std\_logic\_vector(3 downto 0);

test : out std\_logic\_vector(7 downto 0));

end;

architecture arch\_unity of legare\_totala is

signal temp\_break, temp\_make : bit\_vector (7 downto 0) := "11111111";

signal paritate : std\_logic;

signal receptie\_date : std\_logic;

component bistSIreg

port(

CLK\_PS2 : in std\_logic;

D : in bit; -- D acesta reprezinta bit cu bit informatia venita de la tastatura

COD\_TASTA\_ACTUAL : out bit\_vector(7 downto 0);

COD\_TASTA\_PREV : out bit\_vector(7 downto 0);

DATE\_RECEPTIONATE: out std\_logic;

PARITATE\_CORECTA: out std\_logic);

end component;

component Afisare

port( Clock\_placuta: in std\_logic;

STARE, PARITATE: in std\_logic;

IESIRE\_BITI0: in bit\_vector(7 downto 0); -- apelam cu make code

IESIRE\_BITI1: in bit\_vector(7 downto 0); -- apelam cu break code

IESIRE\_TOTALA: out std\_logic\_vector(7 downto 0);

anod\_iesire: out std\_logic\_vector(3 downto 0);

CLK\_ps2: in std\_logic);

end component;

begin

preluare\_cod : bistSIreg port map (CLK\_tastatura, biti\_tastatura, temp\_make, temp\_break, receptie\_date, paritate);

afisare\_placuta : Afisare port map (CLK\_placuta, receptie\_date, paritate, temp\_make, temp\_break, catod\_iesire, anod\_iesire, CLK\_tastatura);

--test <= temp0;

--afis2: SSD port map (biti\_afisor\_segmente,clock\_divizat,catod\_iesire,anod\_iesire);

end;

**5.Lista intrari/iesiri**

Intrări

CLK \_TASTATURA– clock intern

BITI\_TASTATURA– semnal date

CLK\_PLACUTA – clock-ul plăcii

Iesiri

CATOD\_IESIRE – ieșirea principală care va reprezenta caracterul dorit

ANOD \_IESIRE – anozii plăcuței.

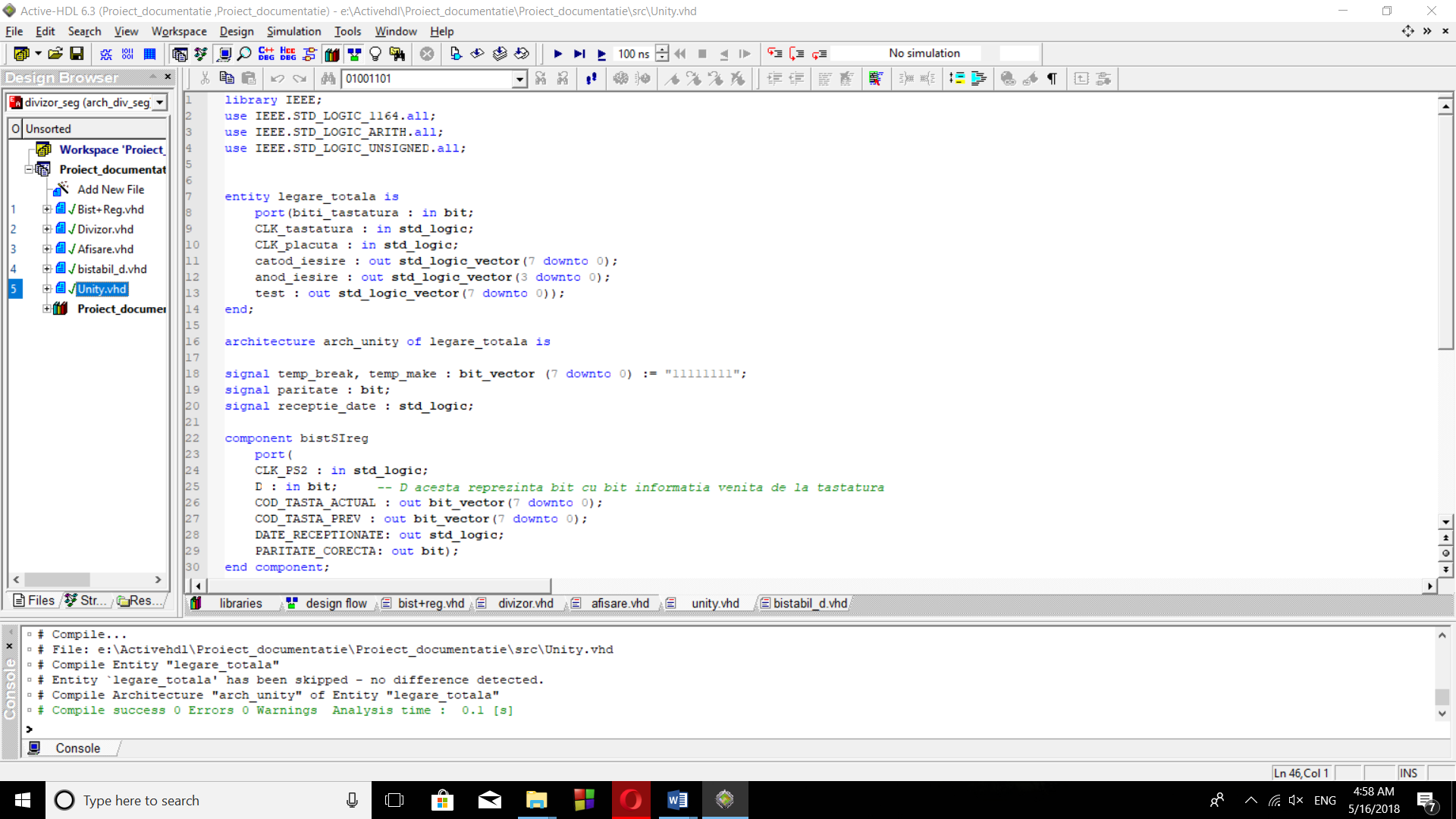
**6.Justificarea soluției alese**

Cunoscand bine modul de transmitere a datelor, ne-am folosit la maxim de transmiterea “break-cod-ului”: cand dam drumul butonului apasat, se genereaza un “F0” + “make-code-ul” tastei apasate. Astfel, la fiecare citire am hotarat sa le retinem ambele, pentru a putea stabili momentul in care shiftam una din tastele deja apasate.

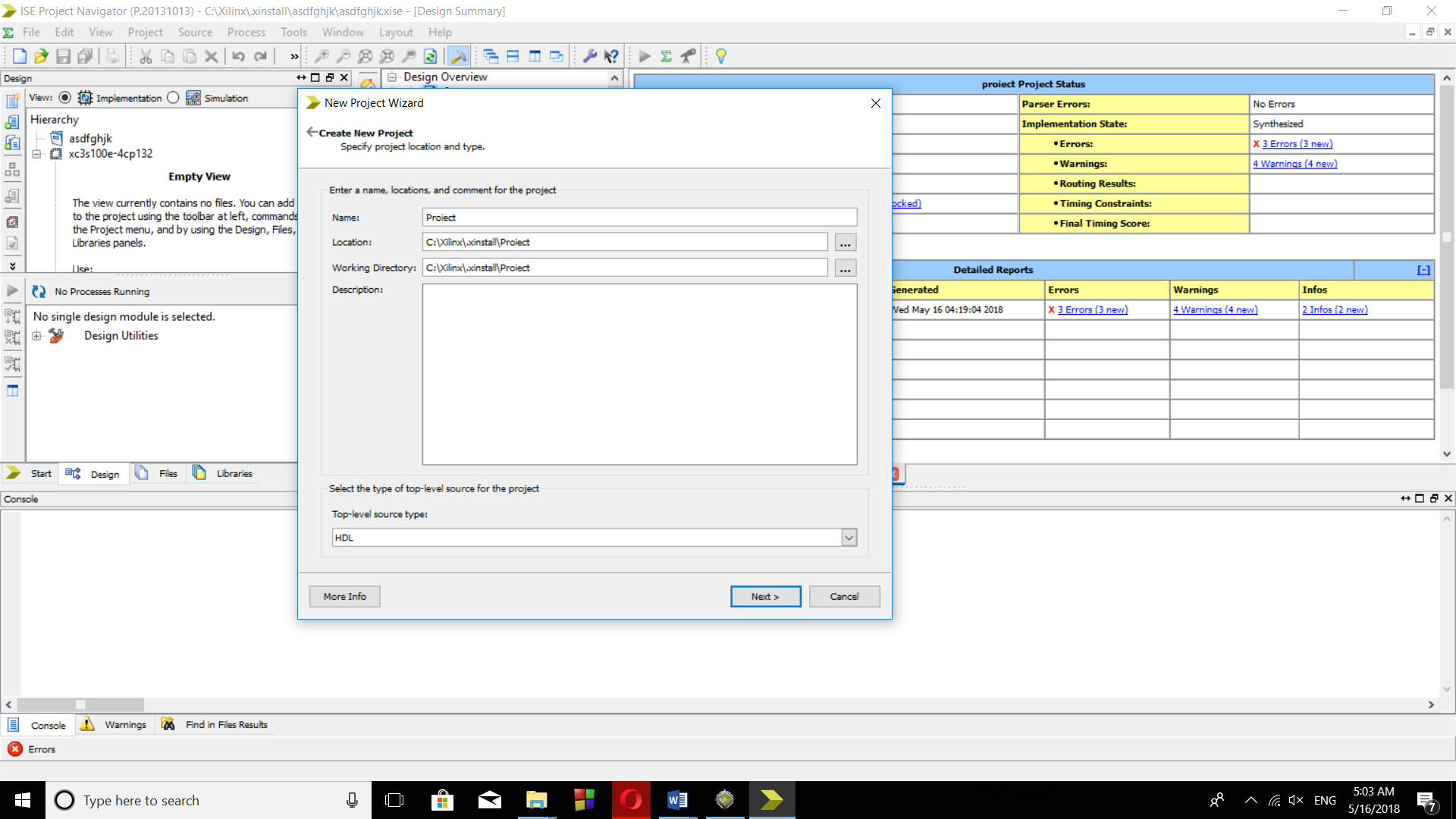
**7.Instrucțiuni de utilizare**

În primul rând trebuie sa prezentăm necesitatea catorva programe pentru a putea crea, compila şi simula codul VHDL şi pentru a putea exporta proiectul pe placa FPGA : ACTIVE-HDL. Modul de utilizare al acestuia din punct de vedere al crearii, complilarii şi simulării codului VHDL. Programul a fost rulat in Xilinx-ISE.

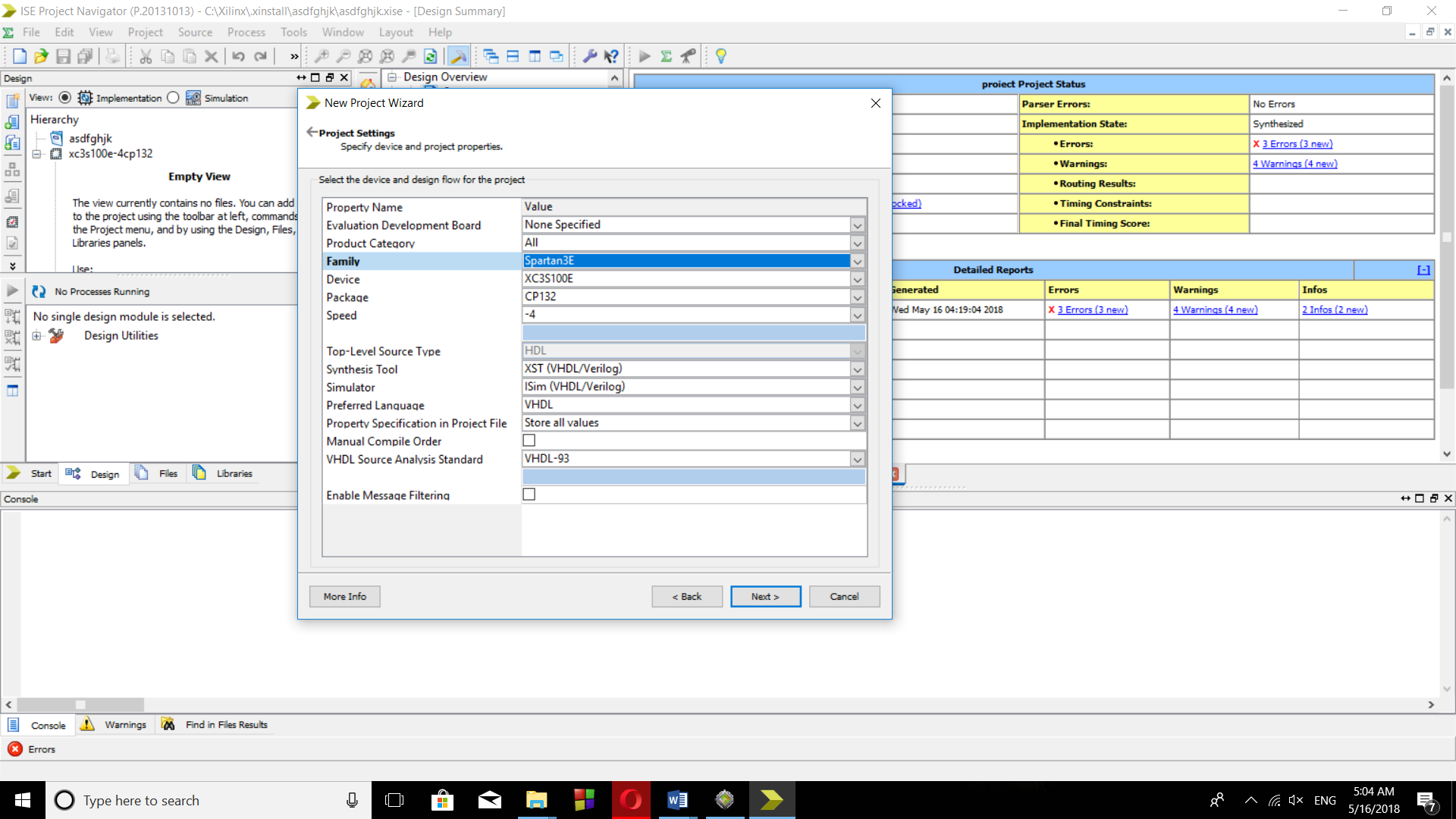
1.Verificarea corectitudinii proiectului in VHDL, pentru a trece la pasul urmator proiectul trebuie sa compileze.



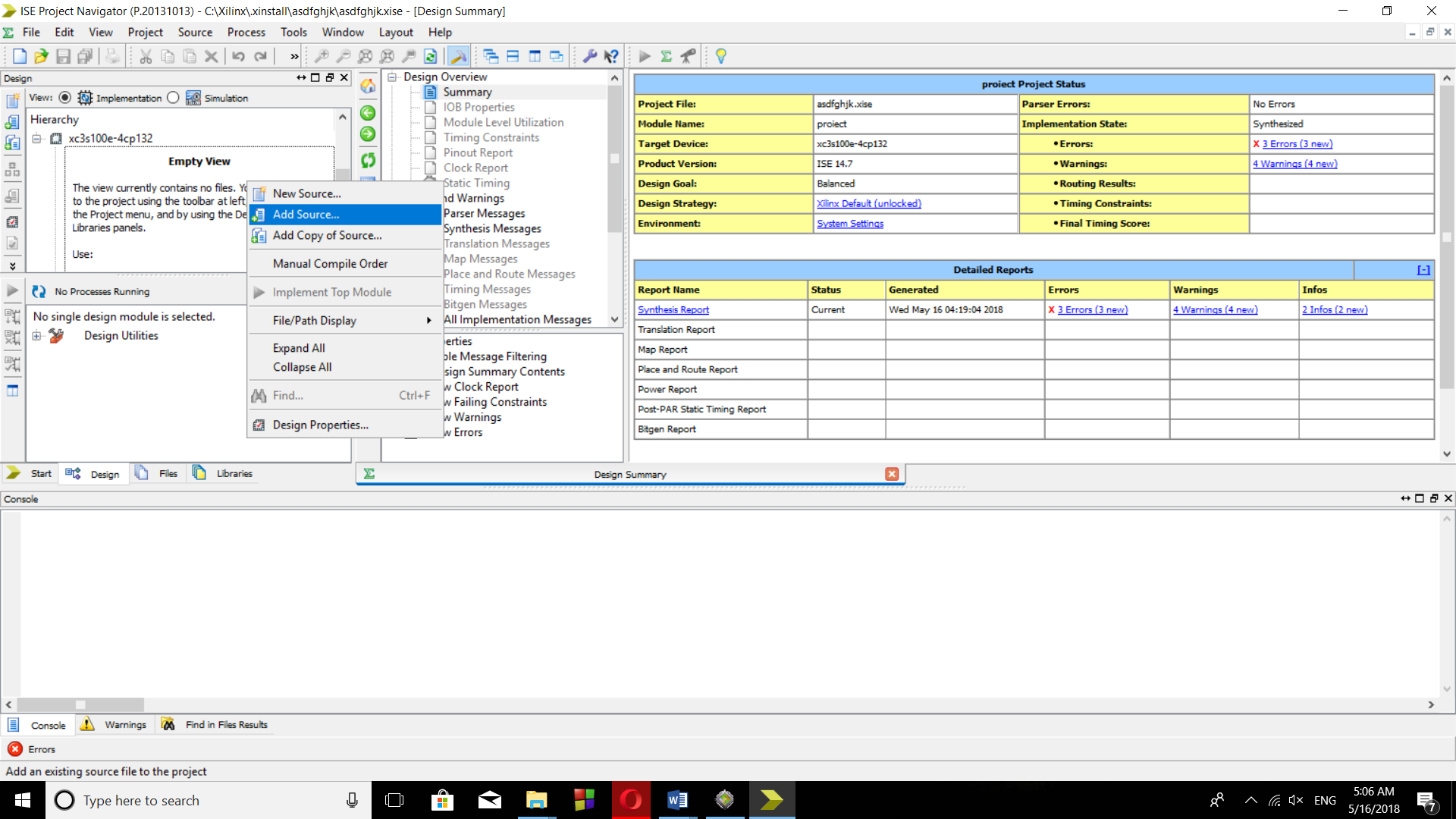
2.Crearea unui proiect in ISE Project Navigator



3.Alegerea specificatiilor placutei



4.Adaugarea fisierelor din VHDL



**8.Posibilitati de dezvoltare ulterioara**

Din punctual de vedere al dezvoltarii ulterioare a proiectului s-ar putea lua in calcul deplasarea punctului zecimal cu ajutorul sagetilor, folosirea tastelor speciale, cum ar fi: Shift, CTRL,CapsLock, NumLock, dar si combinarea mai multor taste. O alta posibila dezvoltare ar fi: prin apasarea CapsLock-ului , placuta sa trimita un semnal inapoi catre tastatura, iar ledul tastei respective sa se aprinda.

Ca “sfat” mai mult decat o posibilitate de dezvoltare ulterioara o reprezinta trecerea de la tastatura PS/2 la tastatura USB intrucat cea din urma a preluat de mult timp “controlul” in ceea ce priveste utilizarea celor doua instrumente hardware.

**9.Bibliografie**

1. <http://www.computer-engineering.org/ps2protocol/>

2 . <http://pcbheaven.com/wikipages/The_PS2_protocol/>

3. https://reference.digilentinc.com/reference/programmable-logic/basys-2/reference-manual