

**CATEDRA DE CALCULATOARE SI TEHNOLOGIA INFORMATIEI**

**Comunicatia dintre placa Nexys4 DDR si o tastatura USB HID**

***Indrumator proiect:*** Lisman Dragos

***Nume student:*** Tusinean Andrei

***Grupa:*** 30236

***Data:*** 16.01.2020

1

**Cuprins**

1. Rezumat – pagina 3
2. Introducere – pagina 4-5
3. Fundamentare teoretica – pagina 6-9
4. Proiectare si implementare – pagina 9-13
5. Rezultate experimentale – pagina 13
6. Concluzii – pagina 13-14
7. Bibliografie – pagina 14
8. Anexe **–** pagina 15

2

**1.Rezumat**

Pentru a lucra cu dispozitive hardware este foarte important sa putem comunica cu ele cat mai eficient , acest lucru poate fi evidentiat usor folosind placa NEXYS4 DDR si o tastatura PS2 .

Pentru dezvoltarea acestui proiect am folosit limbajul vhdl pentru a crea o cale de comunicare intre o placa NEXYS4 DDR si o tastatura cu protocol de comunicare PS2 ,si de asemenea IDE-ul Vivado 2018.3 .

Efectele acestei comunicari se poate remarca pe afisorul de 7 segmente al placutei unde se pot vedea codurile de aparitie si incetare a fiecarei taste apasate .

3

**2.Introducere**

Schimbul de informații și interacțiunea prin o interfață, între două unități de sistem, se realizează prin mărimi fizice sau prin mărimi logice (date) care se pot prezenta sub formă de semnale analogice (continue) sau semnale digitale (discontinue, discrete); un astfel tip de interfata este comunicarea hardware .

Comunicarea hardware reprezinta un aspect foarte important in lumea tehnologiei pentru ca desi un utilizator vede doar parte de front end, o interfata prezentabila , acest lucru nu poate fi posibil fara transmisiile cat de perfecte intre diverse componente hardware .

Desi poate un utilizator nu isi da seama , el defapt realizeaza o comunicare hardware folosind perifericile disponibile : tastatura , mouse , microfon ... Prin utilizarea acestor componente putem interactiona cu sisteme software .

Scopul acestui proiect este de a crea si evidentia o comunicare functionala intre o tastatura cu protocol PS2 si o placuta NEXYS4 DDR , acest lucru putand fi vazut cu ajutorul aifsorului pe 7 segmente al FPGA-ului . Tastatura trimite coduri specifice numite scancodes pentru fiecare tasta , unul care semnaleaza inceperea apasarii tastei , numit makecode , iar altul care semnaleaza incetarea apasarii , numit breakcode . Pe afisor / ssd (seven segment display ) se va afisa makecodul cand se apasa o tasta si breakcode-ul cand se inceteaza apasarea acesteia .

4

Solutia pe care am ales sa o implementez are ca fundament un automat de stari . Automatul trece prin stari in legatura cu ceasul tastaturii , atunci cand a ajuns la numarul dorit de biti cititi ( 11 biti pentru cate un cod ) .

Raportul contine mai multe parti : fundamentare teoretica ,proiectare si implementare , rezultate experimentale , concluzii , bibliografie si anexe .

* + *Sectiunea de Fundamentare teoretica* sunt contine descrierea dispozitivelor, metodelor si toate tehnologiile alese si utilizate pentru realizarea proiectului .
  + *Proiectare si implementare* : este corpul principal al raportului si contine etapele parcurse pentru realizarea proiectului , diagrama de stari , schema bloc si explicatia functionarii proiectului .
  + *Rezultate experimentale :* metodele utilizate pentru testarea proiectului , descrierea implementarii si se specifica limbajul , mediul software si simulatorul .
  + *Concluzia :* reprezinta punctul final al descrierii proiectului si notari despre posibile imbunatatiri .
  + *Bibliografia :* contine o lista a multor resurse utilizate in realizarea proiectului.
  + *Anexa* : contine codul proiectului si lista de makecodes si breakcodes .

5

**3.Fundamentare teoretica**

*3.1 Limbajul si IDE*

Pentru realizarea proiectului am folosit limbajul de descriere hardware VHDL . Este folosit scrie programe care pot fi ulterior testate folosind modele de simulare numite testbench-uri si integrate pe circuite logice pentru diverse sisteme hardware . Cu ajutorul VHDL –ului se poate descrie comportarea sistemului si sa fie verificata / simulata fara a fi nevoie de circuite hardware , toata simularea putan fi realizata in software .

IDE – ul folosit pentru dezvoltarea acestui proiect este Vivado design suite , versiunea 2018.3 care apartine familiei de software Xilinx .Vivado poate fi utilizat pentru a interactiona cu o mare diversitate de FPGA-uri si este o unealta foarte folositoare pentru programarea hardware .

6

*3.2 Nexys4 DDR*

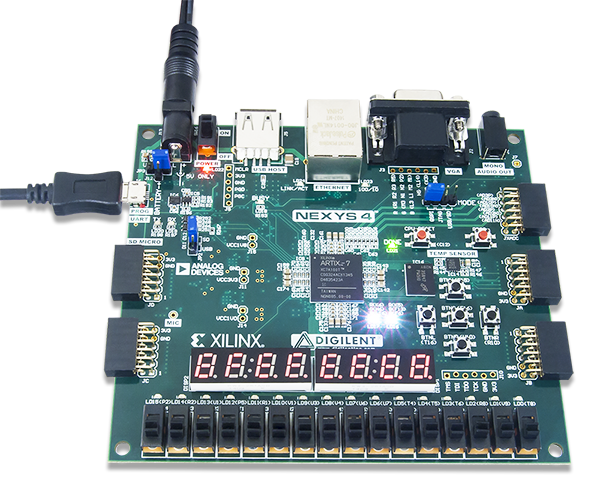


Figura 1

Placa Nexys4 DDR(figura 1) este bazata pe FPGA Artix-7 de la Xilinx. Cu ajutorul sau se pot implementa functionalitati pentru procesoare si pentru circuite combinationale . Are la dispozitie o multitudine de intrari si iesiri pentru o interactionare cat mai eficienta .

7

*3.3 SSD*

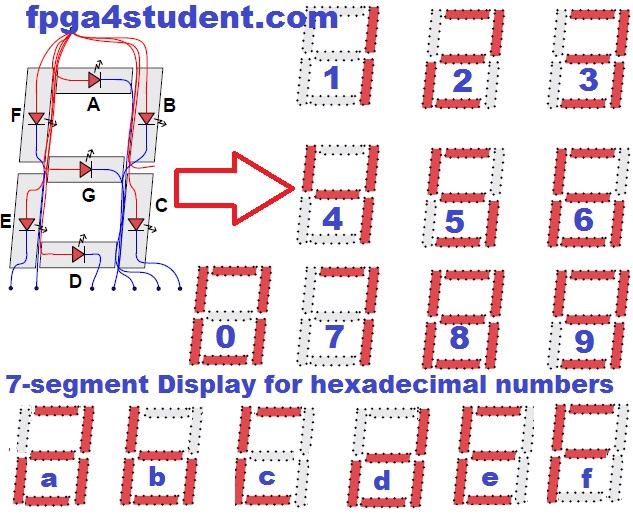


Figura 2 SSD

Este un dispozitiv de iesire folosit de ultizator in cazul acestui proiect pentru a vizualiza makecodurile si breakcodurile specifice tastelor care se apasa. Se vor utiliza 6 din cele 8 afisoare pe 7 segmenete ale placutei NEXYS 4 .

*3.4 Protocolul PS2*

Reprezinta protocul de baza folosit de o tastatura USB HID si de majoritatea mouse-urilor moderne . In cazul tastaturilor cat si al mouse-urilor comunicarea inseamna transmiterea unui de coduri de 11 biti diferite pentru fiecare tasta .

Acest cod consta din 1 bit de start care este 0 , 8 biti care reprezinta codul , un bit de paritate pentru detectare de erori si un bit de stop care trebuie sa fie 1. 8

Tastaturile au un makecode si un breakcode diferite (Figura3)pentru a semnala apasarea , respectiv oprirea apasarii unei taste . Aceasta transmite date pe frontul descrescator al ceasului .

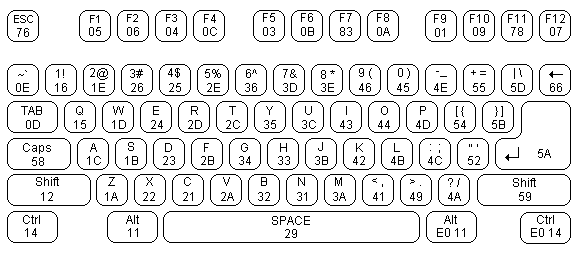
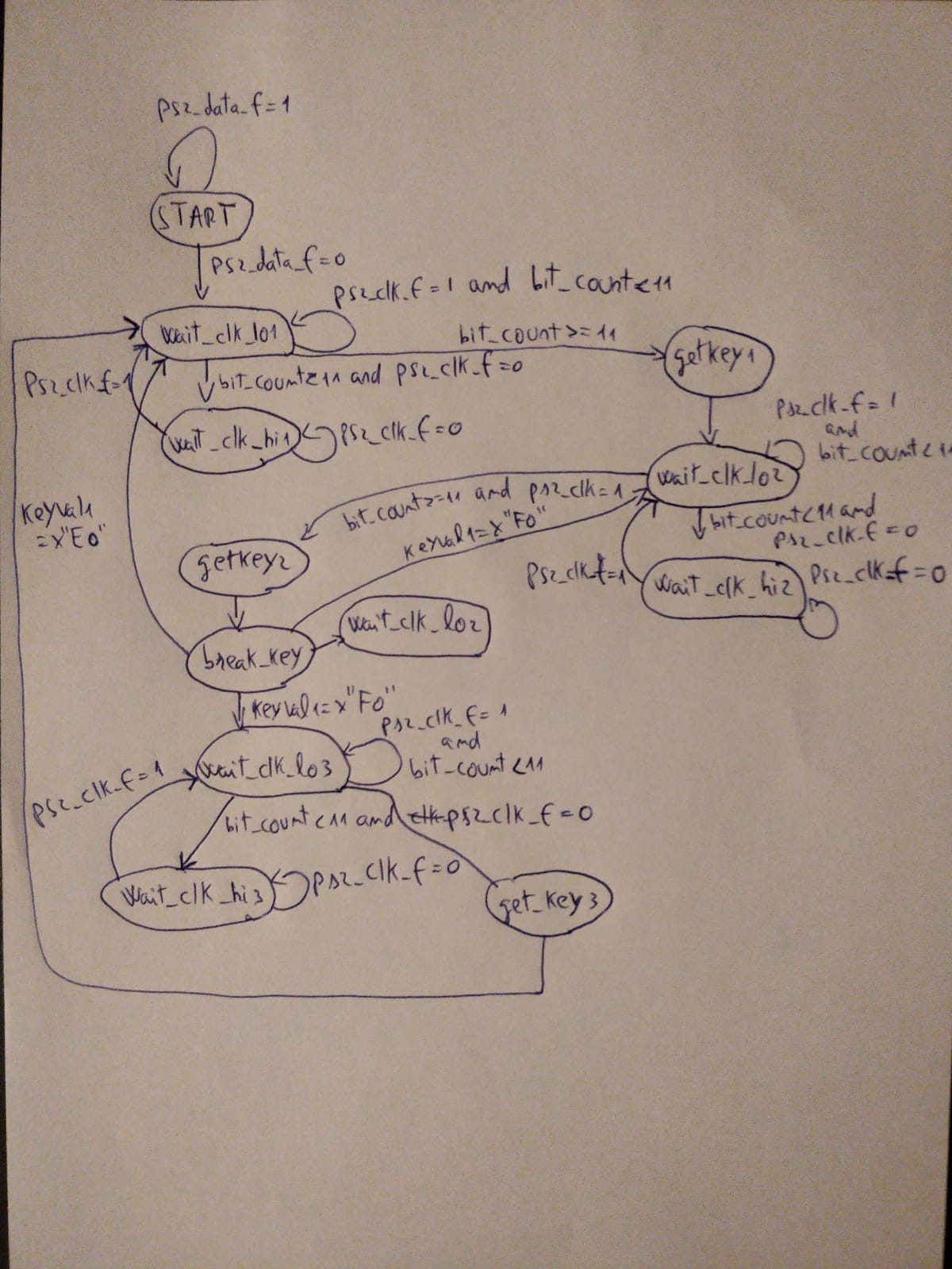


Figura 3

**4.Proiectare si implementare**

*4.1Diagrama de stare :* Figura 4

9



Determinarea schimbarii starilor este data de semnalele ps2\_data\_f , ps2\_clk\_f , bit\_count , keyval1 si keyval2 .

Dupa sincronizare si aplicarea unui debouncer pe ceasul si bitul de date al tastaturi se obtine ps2\_data\_f (bitul de date) si ps2\_clk\_f . Bitul de date este scris pe falling edge (ceasul tastaturii = 0) .

10

Fiecare cod (makecode sau breakcode ) este compus din 11 biti ,care se numara folosint bit\_count care este folosit pe post de numarator pentru a afla cand au fost transmisi 11 biti. In acel moment se scrie in key\_val cei 8 biti care reprezinta codul in sine dupa care contorul este resetat pentru a se putea numara urmatorul cod .

Programul trece prin stari pana iar codurile dorite (keyval1 si 2 ) se vor afisa pe SSD. Keyval1 reprezinta makecodul si keyval2 breakcode-ul , insa daca tasta se tine apasata , cele doua vor reprezenta makecodul pana cand se ridica tasta . Programul reactioneaza la codurile F0 si E0 care reprezinta inceputul unui breakcod .

*4.2 Schema Bloc*

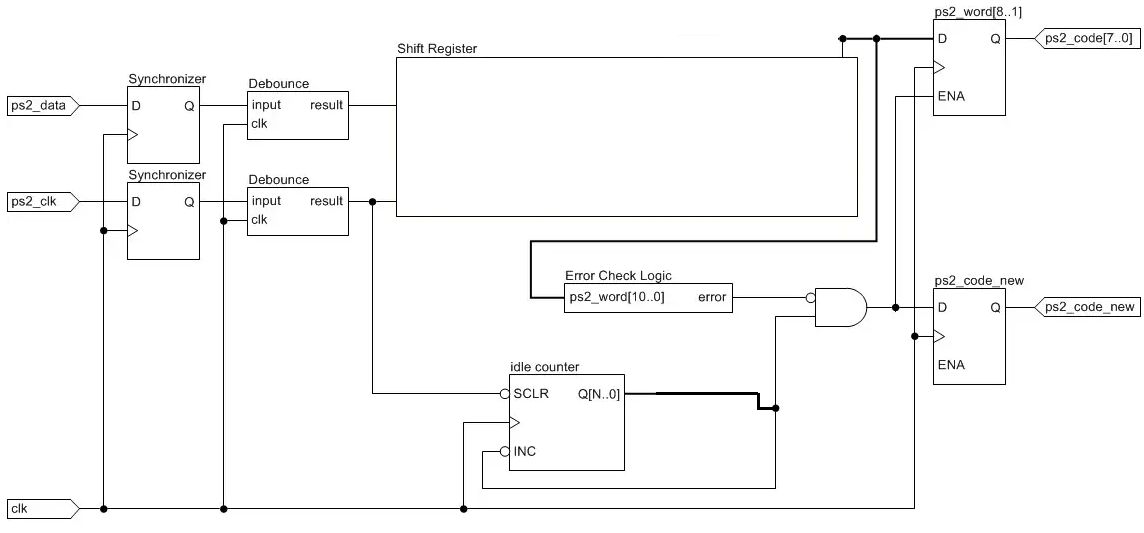


Figura 5

Bitstreamul de date (cate un cod de 11 biti) este sincronizat impreuna cu semnalul de ceas folosind un debouncer pentru a se potrivi cu cele ale placutei dupa care datele sunt scrise in registre comform automatului de stare . Secventele de biti sunt verificate pentru erori care sunt semnalate de un led care se va stinge .

11

**5.Rezultate Experimentale**

Limbajul de descriere hardware folosit este VHDL , impreuna cu IDE-ul Vivado design Suite 2018.3 si placuta FPGA NEXYS4 DDR pentru testarea programului de comunicare cu tastatura pe sistemul de operare Windows 10 .

Placuta utilizata pentru proiect este Nexys4 DDR, care face parte din familia Artix-7 FPGA de la Xilinx. Este optima pentru realizarea comunicarii cu tastarura USB HID pe protocol PS2 . Contine 15850 slice-uri,16 MiB CellularRAM si 123Mib de DDR2 SDRAM, cu frecventa ceasului(semnalului de tact ) de 100 MHz. Pentru realizarea proiectului a fost avut nevoie de portul USB pentru atasarea unei tastaturi USB pe protcol PS2 si a 6 din cele 8 afisoare pe 7 segmente.

Testarea programului se realizeaza doar prin apasarea de taste si analizarea afisoarelor SSD . In momentul in care se apasa o tasta ,pe afisorul cu 7 segmente al placii FPGA apare makecode-ul tastei , iar cand se inceteaza apasarea , apare si break code-ul specific tastei apasate anterior . Am intampinat diverse probleme la partea de transfer a datelor de la tastatura si la afisarea lor pe SSD , aceasta fiind cu fosta versiune a proiectului . Am optat pentru aceasta noua abordare pentru ca cea veche era prea ampla , se baza pe multe module si erau prea multe din ele care puteau sa esueza . Versiunea noua a proiectului , cea finala prezentata , bazata pe un automat de stari este mult mai usor de inteles si este mult mai putin costisitoare din puncte de vedere al cerintelor de procesare a programului .

Partea principala a proiectului o reprezinta automatul de stare care este declarat in fisierul keyboard\_ctrl . Sincronizarea datelor are loc cu ajutorul unui debouncer , dupa care fiind procesate , datele sunt folosite pentru a instantia un SSD pe care se afiseaza rezultatele programului ( codurile corespunzatoare fiecarei taste apasate ) . Datele sunt testate pentru erori folosind un proces care detectare a erorilor care verifica structura secventelor de 11 btit transmisi de la tastatura .

**6.Conculzii**

Scopul proiectului prezentat este evidentierea si implementarea comunicarii dintre o placuta NEXYS4 DDR si o tastatura USB cu protocol PS2 prin oferirea unei semnal vizual care reprezinta aparitia codurilor specifice fiecarei taste apasate , pe afisorul de 7 segmente al placutei FPGA , folosind limbajul de descriere hardware VHDL .

Prin intermediul acestui proiect un utilizator poate vizualiza codurile specifice : make code-ul si break code-ul pentru orice tasta apasata . Break code-ul se afiseaza pe SSD doar atunci cand se inceteaza apasarea unei taste .

12

In cadrul dezvoltarilor ulterioare consider ca s-ar putea implementa diverse actiuni intersesante , cum ar fi detectia de cuvinte , semnale pe led-uri pentru diferite taste precum si interactiunea cu o aplicatie pe calculator folosind tastatura conectata la placuta NEXYS4 DDR . Tot cu aceasta idee s-ar putea dezvolta si un joc care sa poata fi controlat in aceasta maniera (comunicarea hardware dintre cele 2 dispozitive folosite ) .

**7.Bibliografie**

[1] [**https://reference.digilentinc.com/reference/programmable-logic/nexys-4/reference-manual?\_ga=2.252824655.77435359.1571123453-1675411323.1571123453#keyboard**](https://reference.digilentinc.com/reference/programmable-logic/nexys-4/reference-manual?_ga=2.252824655.77435359.1571123453-1675411323.1571123453#keyboard)

[2] <http://users.utcluj.ro/~baruch/ro/pages/cursuri/structura-sistemelor-de-calcul.php>

[3] <http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/ps2/ps2.html>

[4] <https://www.avrfreaks.net/sites/default/files/PS2%20Keyboard.pdf>

[5] <https://wiki.osdev.org/USB_Human_Interface_Devices#Protocol>

13

**8.Anexe**

***8.1 Modulul principal***

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 01/13/2020 12:12:12 PM

-- Design Name:

-- Module Name: keyboard\_ctrl - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

14

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity keyboard\_ctrl is

Port ( clk : in STD\_LOGIC;

clr : in STD\_LOGIC;

ps2Clk : in STD\_LOGIC;

ps2Data : in STD\_LOGIC;

An, Seg: out std\_logic\_vector(7 downto 0);

LED: out std\_logic);

end keyboard\_ctrl;

architecture keyboard\_ctrl of keyboard\_ctrl is

type state\_type is (start, wait\_clk\_lo1, wait\_clk\_hi1, getkey1, wait\_clk\_lo2, wait\_clk\_hi2, getkey2, break\_key, wait\_clk\_lo3, wait\_clk\_hi3, getkey3);

signal state : state\_type;

signal ps2\_clk\_f, ps2\_data\_f : std\_logic; --output debouncer

signal shift1, shift2, shift3 : std\_logic\_vector(10 downto 0); --cele 3 shiftere de 11 biti

signal keyval1, keyval2, keyval3 : std\_logic\_vector(7 downto 0); --valorile de 8 biti din data ale celor 3 shifteri

signal bit\_count : std\_logic\_vector(3 downto 0); --counter de biti

constant cnt\_max : std\_logic\_vector(3 downto 0) := "1011"; --11 --maximul pt counter : 11 biti pentru cate un cod transmis

15

signal data:std\_logic\_vector(31 downto 0);

signal sync\_signals : std\_logic\_vector(1 downto 0);

begin

process(clk)

begin

if rising\_edge(clk) then

sync\_signals(0) <= ps2Clk;

sync\_signals(1) <= ps2Data;

end if;

end process;

D1 : entity WORK.debounce generic map(8) port map(clk, sync\_signals(0), ps2\_clk\_f);

D2 : entity WORK.debounce generic map(8) port map(clk, sync\_signals(0), ps2\_data\_f);

process(clk, clr)

begin

if (clr = '1') then

state <= start;

bit\_Count <= (others => '0');

shift1 <= (others => '0');

shift2 <= (others => '0');

shift3 <= (others => '0');

keyval1 <= (others => '0');

keyval2 <= (others => '0');

keyval3 <= (others => '0');

elsif rising\_edge(clk) then

16

case state is

when start =>

if ps2\_data\_f = '1' then

state <= start;

else

state <= wait\_clk\_lo1;

end if;

when wait\_clk\_lo1 =>

if bit\_Count < cnt\_max then

if ps2\_clk\_f = '1' then

state <= wait\_clk\_lo1;

else

state <= wait\_Clk\_Hi1;

shift1 <= ps2\_data\_f & shift1(10 downto 1);

end if;

else

state <= getkey1;

end if;

when wait\_Clk\_Hi1 =>

if ps2\_clk\_f = '0' then

state <= wait\_Clk\_Hi1;

else

state <= wait\_clk\_lo1;

bit\_Count <= bit\_Count + 1;

end if;

when getkey1 =>

keyval1 <= shift1(8 downto 1);

bit\_Count <= (others => '0');

17

state <= wait\_clk\_lo2;

when wait\_clk\_lo2 =>

if bit\_Count < cnt\_max then

if ps2\_clk\_f = '1' then

state <= wait\_clk\_lo2;

else

state <= wait\_Clk\_Hi2;

shift2 <= ps2\_data\_f & shift2(10 downto 1);

end if;

else

state <= getkey2;

end if;

when wait\_Clk\_Hi2 =>

if ps2\_clk\_f = '0' then

state <= wait\_Clk\_Hi2;

else

state <= wait\_clk\_lo2;

bit\_Count <= bit\_Count + 1;

end if;

when getkey2 =>

keyval2 <= shift2(8 downto 1);

bit\_Count <= (others => '0');

state <= break\_key;

when break\_key =>

if keyval2 = X"F0" then

18

state <= wait\_clk\_lo3;

else

if keyval1 = X"E0" then

state <= wait\_clk\_lo1;

else

state <= wait\_clk\_lo2;

end if;

end if;

when wait\_clk\_lo3 =>

if bit\_Count < cnt\_max then

if ps2\_clk\_f = '1' then

state <= wait\_clk\_lo3;

else

state <= wait\_Clk\_Hi3;

shift3 <= ps2\_data\_f & shift3(10 downto 1);

end if;

else

state <= getkey3;

end if;

when wait\_Clk\_Hi3 =>

if ps2\_clk\_f = '0' then

state <= wait\_Clk\_Hi3;

else

state <= wait\_clk\_lo3;

bit\_Count <= bit\_Count + 1;

end if;

when getkey3 =>

19

keyval3 <= shift3(8 downto 1);

bit\_Count <= (others => '0');

state <= wait\_clk\_lo1;

end case;

end if;

end process;

Data(23 downto 0) <= keyval3 & keyval2 & keyval1;

process

variable cnt : INTEGER := 0;

begin

LED <= '0'; --Ca default datele sunt corecte

if shift1(0) = '1' OR shift1(10) = '0' then --start bit trebuie sa fie 0 si stop 1

LED <= '1';

else

for i in 1 to 8 loop

if shift1(i) = '1' then

cnt := cnt + 1;

end if;

end loop;

if ((cnt mod 2) = 0 AND shift1(9) = '0') OR ((cnt mod 2) = 1 AND shift1(9) = '1') then

LED <= '1'; --daca avem nr par de 1 atunci parity bit trebuie sa fie 1

end if; --daca avem nr imoar de 1 atunci parity bit trebuie sa fie 0

end if;

end process;

20

sevenSeg: ENTITY WORK.displ7seg port map(clk, clr, Data, An, Seg);

end keyboard\_ctrl;

***Afisorul pe 7 segmente***

----------------------------------------------------------------------------------

-- Company: UTCN

-- Engineer:

--

-- Create Date: 02/04/2016 10:12:56 AM

-- Design Name: displ7seg

-- Module Name: displ7seg - Behavioral

-- Project Name:

-- Target Devices: Nexys4 DDR (xc7a100tcsg324-1)

-- Tool Versions: Vivado 2015.4, Vivado 2016.4

-- Description: Multiplexor pentru afisajul cu 7 segmente

-- Datele de la intrare se interpreteaza ca valori hexazecimale

-- si sunt decodificate in configuratia segmentelor afisajului

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

21

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

use IEEE.STD\_LOGIC\_ARITH.all;

entity displ7seg is

Port ( Clk : in STD\_LOGIC;

Rst : in STD\_LOGIC;

Data : in STD\_LOGIC\_VECTOR (31 downto 0); -- datele pentru 8 cifre (cifra 1 din stanga: biti 31..28)

An : out STD\_LOGIC\_VECTOR (7 downto 0); -- selectia anodului activ

Seg : out STD\_LOGIC\_VECTOR (7 downto 0)); -- selectia catozilor (segmentelor) cifrei active

end displ7seg;

architecture Behavioral of displ7seg is

constant CNT\_100HZ : integer := 2\*\*20; -- divizor pentru rata de reimprospatare de ~100 Hz (cu un ceas de 100 MHz)

signal Num : integer range 0 to CNT\_100HZ - 1 := 0;

signal NumV : STD\_LOGIC\_VECTOR (19 downto 0) := (others => '0');

signal LedSel : STD\_LOGIC\_VECTOR (2 downto 0) := (others => '0');

signal Hex : STD\_LOGIC\_VECTOR (3 downto 0) := (others => '0');

begin

-- Proces pentru divizarea ceasului

divclk: process (Clk)

Begin

22

if (Clk'event and Clk = '1') then

if (Rst = '1') then

Num <= 0;

elsif (Num = CNT\_100HZ - 1) then

Num <= 0;

else

Num <= Num + 1;

end if;

end if;

end process;

NumV <= CONV\_STD\_LOGIC\_VECTOR (Num, 20);

LedSel <= NumV (19 downto 17);

-- Selectia anodului activ

An <= "11111110" when LedSel = "000" else

"11111101" when LedSel = "001" else

"11111011" when LedSel = "010" else

"11110111" when LedSel = "011" else

"11101111" when LedSel = "100" else

"11011111" when LedSel = "101" else

"10111111" when LedSel = "110" else

"01111111" when LedSel = "111" else

"11111111";

-- Selectia cifrei active

Hex <= Data (3 downto 0) when LedSel = "000" else

Data (7 downto 4) when LedSel = "001" else

23

Data (11 downto 8) when LedSel = "010" else

Data (15 downto 12) when LedSel = "011" else

Data (19 downto 16) when LedSel = "100" else

Data (23 downto 20) when LedSel = "101" else

Data (27 downto 24) when LedSel = "110" else

Data (31 downto 28) when LedSel = "111" else

X"0";

-- Activarea/dezactivarea segmentelor cifrei active

Seg <= "11111001" when Hex = "0001" else -- 1

"10100100" when Hex = "0010" else -- 2

"10110000" when Hex = "0011" else -- 3

"10011001" when Hex = "0100" else -- 4

"10010010" when Hex = "0101" else -- 5

"10000010" when Hex = "0110" else -- 6

"11111000" when Hex = "0111" else -- 7

"10000000" when Hex = "1000" else -- 8

"10010000" when Hex = "1001" else -- 9

"10001000" when Hex = "1010" else -- A

"10000011" when Hex = "1011" else -- b

"11000110" when Hex = "1100" else -- C

"10100001" when Hex = "1101" else -- d

"10000110" when Hex = "1110" else -- E

"10001110" when Hex = "1111" else -- F

24

"11000000"; -- 0

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 01/13/2020 12:39:36 PM

-- Design Name:

-- Module Name: debounce - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

25

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity debounce is

generic(

cnt\_size : INTEGER := 19);

Port ( clk : in STD\_LOGIC;

btn : in STD\_LOGIC;

result : out STD\_LOGIC);

end debounce;

architecture Behavioral of debounce is

signal flipflops : std\_logic\_vector(1 downto 0); --input

signal cnt\_set : std\_logic; --reset

signal cnt\_out : std\_logic\_vector(19 downto 0) := (others => '0');

begin

cnt\_set <= flipflops(0) xor flipflops(1);

process(clk)

begin

if(clk'event and clk = '1') then

26

flipflops(0) <= btn;

flipflops(1) <= flipflops(0);

if(cnt\_set = '1') then

cnt\_out <= (others => '0');

elsif(cnt\_out(cnt\_size) = '0') then

cnt\_out <= cnt\_out + 1;

else

result <= flipflops(1);

end if;

end if;

end process;

end Behavioral;

27