Universitatea Tehnică din Cluj-Napoca

Catedra de Calculatoare și Tehnologia Informației

Comunicația dintre o placă Basys3 și o tastatură USB

Student: Trifu Diana Maria

Grupa: 30233

Îndrumător de proiect: Fati Daniela

Data: 4 ianuarie 2020

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Cuprins

1. Rezumat

Cred că fiecare dintre noi s-a întrebat măcar odată: Cum știe calculatorul nostru ce tastă a tastaturii a fost apăsată? Cred că acest proiect a putut să ofere un răspuns foarte clar și o perspectivă amănunțită asupra procesului care face posibil acest lucru. Etapele parcurse în atingerea obiectivului final, au putut lămurii toate curiozitățile legate de acest aspect. Având ca scop final afișarea unei taste apăsate, pe un anod al plăcii Basys3, acest proiect a avut ca obiectiv și stabilirea unei conexiunii între tastatură și placă.

Soluția proiectului a fost concepută cu ajutorul a patru module scrise în limbajul VHDL. Cele patru module au fost concepute în scopul realizării conexiunii dintre cele două componente, făcând astfel posibilă transmiterea datelor de la tatstatură la placă și convertirea acestor date într-o succesiune de opt biți, făcându-se astfel posibilă afișarea caracterelor corespunzătoare tastelor afișate, pe prima cifră din partea dreaptă a afișorului plăcii.

La final, în urma încărcării pe placă a bitstream-ului generat în urma sintezei și implementării celor patru module, alături de constrângerile corespunzătoare plăcii, a fost vizibil că pe primul anod al plăcii apărea reprezentarea caracterului corespunzător tastei apăsate. La apăsarea anumitor caractere, a căror reprezentare nu a fost posibilă, va apărea o singură linie, aceasta sugerând lipsa posibilității de reprezentare a acelui caracter.

2. Introducere

Conform celei mai largi și generoase accepțiuni, calculatorul poate fi privit ca un sistem ce primește la intrare un flux de date asupra cărora operează prelucrările indicate de utilizator, oferind rezultatele acestora sub forma unui flux informațional emergent. Transferul de informație între mediul extern și mașina de calcul trebuie să țină cont de particularitățile fiecăreia dintre cele două entități, în special în ceea ce privește accesul la semnificația asociată fluxurilor de intrare/ieșire. În acest context, noțiunea de reprezentare a informației capătă două accepțiuni complementare:

- **reprezentarea internă, ce reflectă de fapt modul de organizare a sistemului de calcul**
- **reprezentarea externă, asociată utilizatorului**

Între cele două tipuri de reprezentări există și o diferență de nuanță: în timp ce prima este caracterizată de rigurozitate(fiind impusă de organizarea structurală a calculatorului), cea de-a doua are pronunțate trăsături subiective(semnificația unei entități informaționale este acordată de utilizator însuși, în conformitate cu propria sa percepție). [2]

În cadrul acestei domeniu de studiu (Structura Sistemelor de Calcul) este propusă analiza prezentării generale a modului de organizare internă a unui calculator. Pentru a putea utiliza corect această platformă – numită generic sistem de calcul – trebuie însușită o manieră pertinentă a noțiunilor de bază referitoare la structura sa constructivă (hardware) precum și la programarea lor(prin dezvoltare software). Programatorii au nevoie de aceste informații constructiv-funcționale pentru implementarea de o manieră optimală, sub aspectul performanțelor, a unei aplicații de date.

Un sistem de calcul este caracterizat de structura sa, această noțiune identificând practic totalitatea componentelor individuale și a subansamblurilor ce sunt necesare asigurării funcțiilor specificate. Din punct de vedere obiectiv, semnificația „structurii" este preponderent orientată către accepțiunea fizică, elementele structurale de bază fiind extrem de „palpabile": circuite integrate, componente electronice discrete, căi de semnal, sursă de alimentare cu tensiune.

Obiectivul acestui proiect este realizarea cu succes a unei conexiuni între o placă Basys3(placa Basys3 este o platformă completă de dezvoltare a circuitelor digitale, bazată pe cel mai recent Artix-7 (FPGA) de la Xilinx) și o tastatură cu port USB. Se dorește ca în momentul în care o tastă de la tastatură va fi apăsată, aceasta să fie afișată pe unul dintre afișoarele de 7 segmente de care dispune placa.

Proiectarea este realizată în limbajul VHDL (Very High Description Language) de descriere a hardware-ului. Proiectul este creat în mediul de dezvoltare Vivado 2019.2 Design Suite de la Xilinx.

Acesta este alcătuit din doar patru componente, fiecare dintre acestea cu un rol foarte bine stabilit, contribuind la atingerea obiectivului final. Problema pe care o propune acest proiect a fost împărțită în mai multe subprobleme, astfel fiind nevoie de 3 componente: câte o componentă pentru fiecare subproblemă: realizarea conexiunii dintre placa Basys3 și tastatură prin intermediul interfeței PS/2, filtrarea oscilațiilor unei taste și afișarea caracterelor pe afișajul cu șapte segmente al plăcii.

În ceea ce urmează vor fi furnizate mai multe detalii în ceea ce implică acest proiect din toate punctele de vedere. Prima dată vă vor fi prezentate elementele de fundamentare teoretică, cum ar fi: modelele, metodele și tehnologiile utilizate pentru atingerea obiectivului principal. În continuare, în partea principală a acestui raport, va fi prezentată descrierea fiecărei etape parcurse pentru realizarea proiectului. Descrierea acestor etape va fi urmată de rezultatele simulării și interpretarea acestora. În încheiere, în ultima secțiune vor fi discutate aspecte legate de procesul de realizare a proiectului, dar și posibile dezvoltări ulterioare a acestuia.

3. Fundamentare teoretică

3.1. Placa Basys3

Această placă este o platformă completă de dezvoltare a circuitelor digitale, bazată pe cel mai recent Artix-7 (FPGA) de la Xilinx. Cu FPGA de mare capacitate (numărul piesei Xilinx XC7A35T-1CPG236C), costul general redus și colecția de porturi USB, VGA și alte porturi, Basys3 poate găzdui proiecte variând de la circuite combinaționale introductive până la circuite secvențiale complexe, cum ar fi procesoare și controlere încorporate. Include suficiente comutatoare, LED-uri și alte dispozitive I/O pentru a permite finalizarea unui număr mare de proiecte fără a fi nevoie de hardware suplimentar și suficienți pini I/O FPGA. [1]

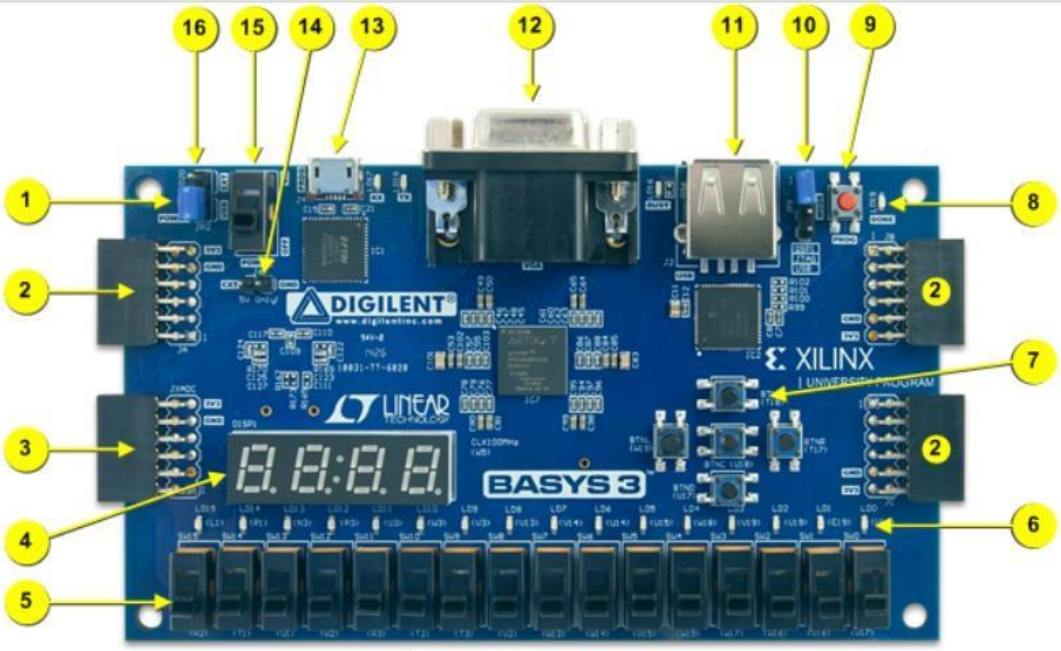
Elemente:

Artix-7 FPGA este optimizat pentru logica de înaltă performanță și oferă mai multă capacitate, performanță mai mare și mai multe resurse decât proiectele anterioare. Caracteristicile Artix-7 35T includ:

- **33,280 de celule logice în 5200 de bucăți (fiecare bucată conține patru 6-intrari LUTs și 8 bistabile);**
- **1,800 KBITS RAM;**
- **Cinci dispozitive de gestionare a ceasului, fiecare cu o bucla de blocare (PLL);**
- **90 DSP bucăți;**
- **Vitezele interne ale ceasului depășesc 450MHz;**
- **Convertor analog-digital pe cip (XADC).**

Această placă oferă si colecție îmbunătățită de porturi și periferice, incluzând:

- **16 comutatoare**
- **16 LED-uri**
- **5 butoane**
- **Afișaj 4-cifre 7-segmente**
- **3 porturi Pmod**
- **Pmod pentru semnale XADC**
- **Ieșire VGA pe 12 biți**
- **Podul USB-UART**
- **Serial Flash**
- **Portul Digilent USB-JTAG pentru programarea și comunicarea FPGA**
- **USB HID Host pentru mouse-uri, tastaturi sau stick-uri de memorie**



Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Figura 3.1.1. Prezentarea plăcii Basys3

3.2. Host-ul USB

În informatică, clasa de dispozitive de interfață umană USB (clasa USB HID) face parte din specificația USB pentru periferice de computer: specifică o clasă de dispozitiv (un tip de hardware de computer) pentru dispozitive de interfață umană, cum ar fi tastaturi, mouse-uri, controlere de joc dispozitive de afișare alfanumerice.

După pornire, fie se descarcă un flux de biți pe FPGA, fie se așteaptă să fie programat din alte surse. Odată ce FPGA este programat, microcontrolerul trece în modul aplicație, care, în acest caz, este modul USB HID Host. Firmware-ul din microcontroler poate conduce un mouse sau o tastatură atașată la conectorul USB de tip A la J2 etichetat „USB". Suportul Hub nu este disponibil în prezent, deci poate fi utilizat doar un singur mouse sau o singură tastatură. Interfață standard pentru comunicarea cu mouse-ul sau tastatura este PS/2. [1]

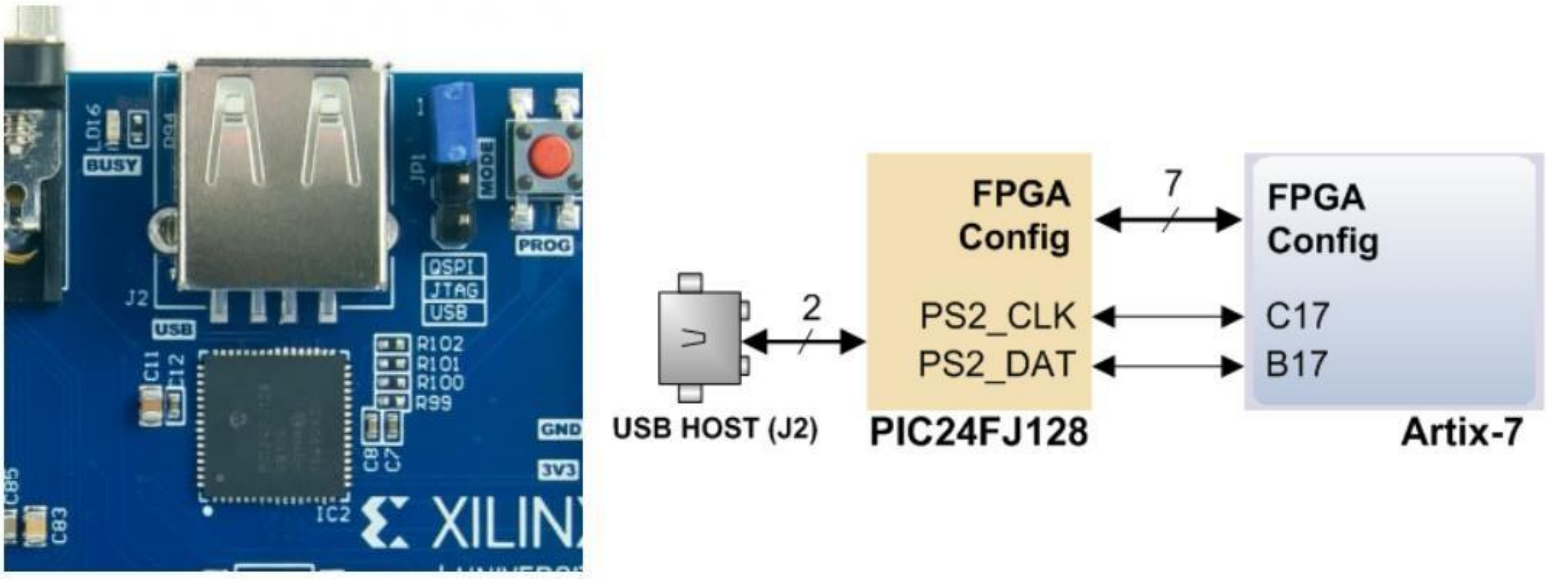


Figura 3.2.1. Transmiterea datelor către placă prin portul USB

3.3. Tastatura

Tastaturile tip PS/2 folosesc coduri de scanare pentru a comunica informație despre apăsarea tastelor. Fiecărei taste îi este asignat un cod de scanare care este transmis oricând aceasta este apăsată. Dacă o tastă se menține apăsată, codul de scanare va fi transmis în mod repetat, o dată la aproximativ 100ms. Când o tastă este eliberată, se transmite un cod de tastare F0, urmat de codul de scanare al tastei eliberate. [1]

Codurile de scanare ale tastelor sunt prezentate în figura de mai jos:

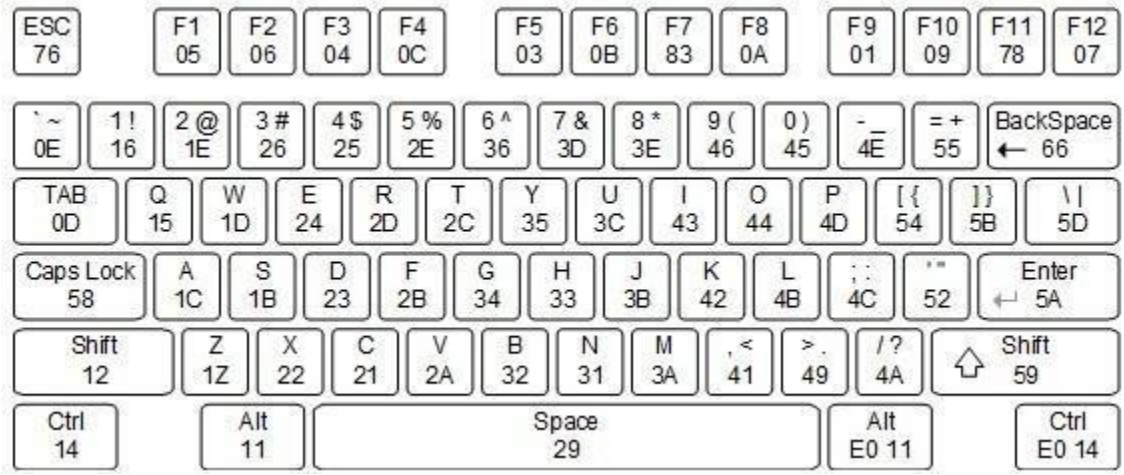


Figura 3.3.1. Codurile de scanare ale tastaturii

3.4. Afișor de 7 segmente

Placa Basys3 conține un afișaj LED(7-segmente) cu patru cifre cu anod comun. Fiecare cifră este compusa din câte 7 segmente aranjate precum în figura de mai jos, fiecare segment având un LED încorporat. LED-urile de segment pot fi illuminate individual, astfel încât orice model din cele 128 poate fi afișat, unele LED-uri din segmente fiind aprinse, iar celelalte stinse. Pentru a lumina un segment anodul trebuie sa fie H (high), iar catodul L(low). Pentru ca cele 4 cifre disponibile să creeze iluzia de a fi aprinse simultan, acestea trebuie setate o data la 1-16ms, la o frecvență de la 1 KHz la 60 KHz.[1]

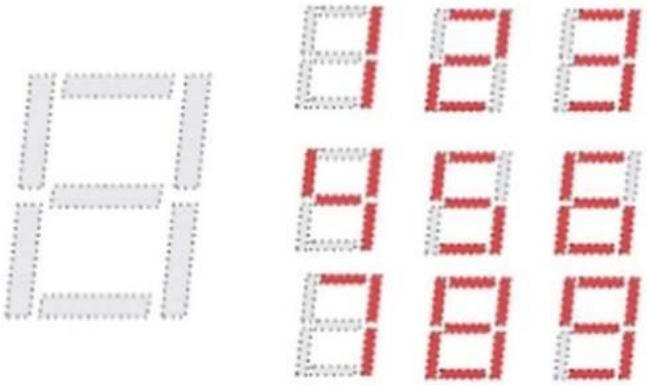


Figura 3.4.1. Reprezentarea cifrelor pe un anod al plăcii

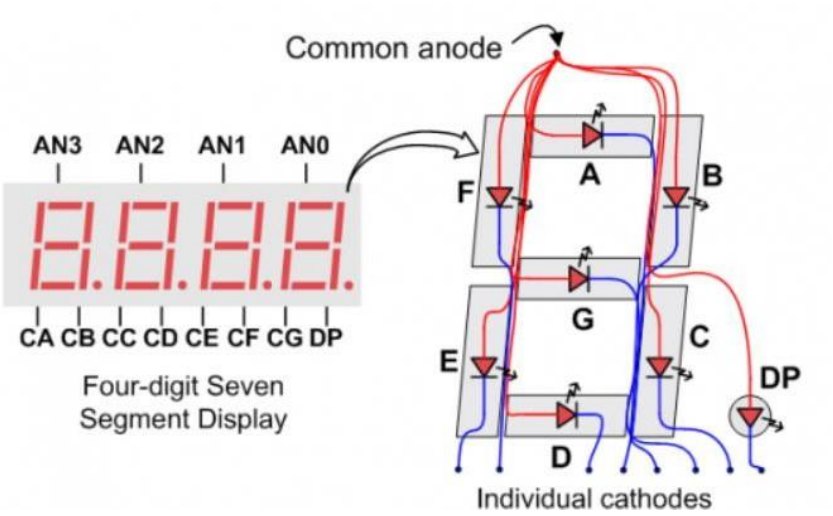


Figura 3.4.2. Prezentarea unei cifre a afișajului și a celor 7 segmente

4. Proiectare și implementare

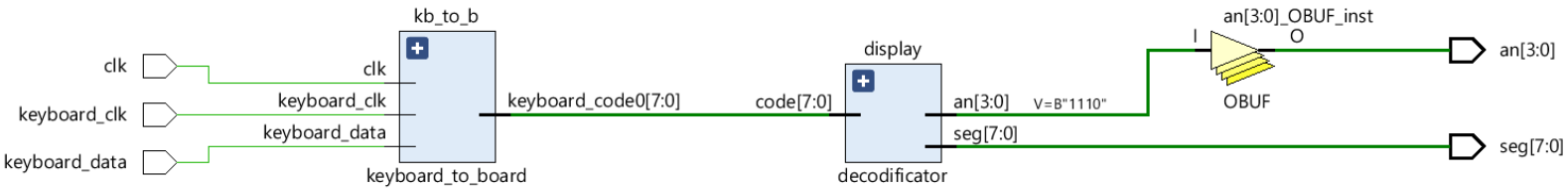
4.1. Soluția aleasă

Am ales să analizez inițial problema propusă și să o divizez în cât mai multe subprobleme, astfel încât să pot atribui fiecare subproblemă unei componente. Având ca și obiectiv final afișarea unor caractere, a căror taste au fost apăsate pe o tastatură conectată la placa Basys3, am analizat teoretic ce ar trebui să se întâmple la fiecare pas: placa trebuie să preia corespunzător datele transmise de către tastatură atunci când o tastă este apăsată. Mai apoi, datele pe care aceasta le primește ar trebui să fie convertite într-un format care să faca posibilă afișarea caracterelor pe afișorul cu 7 segmente.

Având ca și punct de pornire aceste două etape importante de parcurs pentru a atinge obiectivul final, am început prin implementarea componentei „keyboard_to_board.vhd" care asigură conectarea tastaturii la placa Basys3, prin intermediul interfeței PS/2. Cel de-al doilea modul care a fost implementat, „decodificator.vhd", se ocupă de conversia codurilor primite de la tastele apăsate ale tastaturii într-o secvență de 8 biți care reprezintă cifra afișorului cu 7 segmente pe care urmează să fie afișat caracterul respectiv. Cel de-al treilea modul implentat, „debouncer.vhd", este un filtru de oscilații pentru butoanele tastaturii. Această componentă a fost necesară deoarece apăsarea unui buton poate genera adesea unele tranziții „false" din cauza unor probleme fizice și mecanice, datele transmise de apăsarea butonului fiind extrem de imprevizibile. Aceste tranziții pot fi interpretate greșit, spre exemplu: apăsări succesive ale tastei, acest lucru producând rezultate eronate în cadrul programului implementat. Această componentă este folosită în cadrul componentei „keyboard_to_board". Ultima componentă implementată este „modulul_principal.vhd" care conectează cele doua compoenente principale: „keyboard_to_board.vhd" și „decodificator.vhd", împreună cu anozii și catozii plăcii pe care urmează a fi afișate caracterele care corespund tastelor apăsate de utilizator.

Am ales această implementare întrucât în spatele acesteia se află o bună logică și analiză a problemei propuse. Codul propus este succint și ușor de înțeles de către orice persoană familiarizată cu aceste noțiuni și acest limbaj de programare hardware.

4.2. Arhitectura generală



4.3. Detalii de implementare

Keyboard_to_board.vhd

Semnale de intrare:

clk: in STD_LOGIC
keyboard_clk: in STD_LOGIC
keyboard_data: in STD_LOGIC

Semnale de ieșire:

keyboard_code0: out STD_LOGIC_VECTOR(7 downto 0)
keyboard_code1: out STD_LOGIC

Majoritatea tastaturilor au fost fabricate astfel încât să fie compatibile, atât cu interfața USB, cât și interfața PS/2. Astfel, această componentă se folosește de interfața PS/2 pentru a conecta cele două componente hardware, permițând transmiterea datelor dinspre tastatură, înspre placă. Două dintre cele mai importante semnale de intrare ale acestei componente (keyboard_clk, keyboard_data) sunt de tip STD_LOGIC și reprezintă semnalele de ceas și de date ale tastaturii. Acestea două sunt prima dată filtrate cu ajutorul componentei „debouncer.vhd" și sincronizate. Mai apoi semnalul intern de date este încărcat serial într-un registru de shiftare , deplasarea având loc pe nivelul logic '0' al semnalului de ceas al sistemului(clk, de tip STD_LOGIC).

În continuare este descrisă logica de verficare a erorilor combinaționale care ar putea apărea, verificând biții de date cu bitul de paritate, bitul de început și cel de oprire. Ultimul proces descris în această componentă este cel care stabilește momentul în care schimbul de date a fost realizat, definit de semnalul de ceas al tastaturii. În momentul în care contorul a ajuns la o valoare egală cu raportul dintre frecevența semnalului de ceas a sistemului și 18_000, care semnifică faptul că semnalul de ceas al tastaturii a avut valoarea logică '1' o perioadă mai scurtă de timp decât jumătate din perioada de ceas a sistemului, și nicio eroare nu a fost detectată putem seta un indicativ(keyboard_code0) care semnifică faptul ca acel cod transmis de apăsarea unei taste este disponibil, iar semnalul de ieșire „keyboard_code0" primește noul cod recepționat de la tastatură.

Decodificator.vhd

Semnale de intrare:

code: in STD_LOGIC_VECTOR(7 downto 0)

Semnale de ieșire:

an: out STD_LOGIC_VECTOR(3 downto 0)
seg: out STD_LOGIC_VECTOR(7 downto 0)

Această componentă are în componența sa descris un singur process, care conține o structură „case…end case". Acest process are în lista sa de sensibilități semnalul de intrare pe 8 biți: „code". Acest semnal reprezintă codul din spatele tastei apăsate care a fost transmis de la tastatură către placa Basys3 prin intermediul interfeței PS/2. De fiecare data când codul preluat de placă își modifică valoarea, acesta va fi trecut printr-un process de decodificare în cadrul acestei componente. În cadrul acestui proces, în funcție de codul receptat și de decizia de design a carcaterului luată de dezvoltator, se va decide pentru fiecare segment al anodului ales dacă va fi aprins sau stins. Datorită limitării impuse de modul de distribuție a catozilor unui anod, unele litere nu pot fi reprezentate sugestiv, iar altele ar putea fi reprezentate, dar reprezentarea lor ar fi confundată cu a altora, sau o anumită reprezentare s-ar regăsi în cazul a două caractere, creându-se o confuzie, așa că am ales ca atunci când una din tastele care corespund caracterelor: G, K, M, S, T, W, X și Z, este apasată să fie reprezentată prin aprinderea unui singur catod: „g". Reprezentarea tuturor caracterelor se regăește în „Anexa F".

Asignarea de la finalul arhitecturii acestei componente, a fost realizată luând decizia de a reprezenta toate aceste carcatere pe prima cifră a afișorului din partea dreaptă.

Debouncer.vhd

Semnale de intrare:

clk: in STD_LOGIC
din: in STD_LOGIC

Semnale de ieșire:

qout: out STD_LOGIC

Această componentă ajuta la filtrarea oscilațiilor intrarilor de semnal de ceas si cea de date ale componentei „keyboard_to_board.vhd". Această componentă are ca și intrări doua semnale de tip STD_LOGIC, una dintre ele reprezentând semnalul de ceas, iar cealaltă semnalul care trebuie stabilizat și preluat corect dupa apăsarea unui buton. Ieșirea componenti este reprezentată de un semnal de tip STD_LOGIC care reprezinta semnul de intrare stabilizat, preluat corect. Este vorba despre doua bistabile, iesirea primului bistabil reprezentând intrarea pentru cel de-al doilea bistabil. Cele două memorează ultimele două nivele logice ale butonului, dar cand valorile pe ieșire ale acestor două bistabile sunt identice, atunci intervine problema, care această componentă dorete să o rezolve, folosindu-se de un numărător și o poartă XOR. Operația de XOR între ieșirele celor două bistabile verifică dacă ieșirile lor sunt egale, iar în caz afirmativ declanșează numărătorul. Contorul crește până reușește să atingă limita impusă și să seteze semnalul de ieșire sau se întrerupe și îi este șterasă valoarea de către rezultatul operației de XOR dintre ieșirile bistabilelor, deoarece valoarea logică a butonului nu a fost încă stabilizată.

Modulul_principal.vhd

Semnale de intrare:

clk : in STD_LOGIC
keyboard_clk : in STD_LOGIC
keyboard_data : in STD_LOGIC

Semnale de ieșire:

an : out std_logic_vector (3 downto 0)
seg : out std_logic_vector (7 downto 0)

Acest modul face legătura între componentele: „keyboard_to_board.vhd”, „decodificator.vhd” și afișor plăcii Basys3. Această componentă conține două mapări generice ale celor două componente menționate anterior.

5. Rezultate experimentale

Pentru a putea demonstra funcționalitatea acestui proiect a fost nevoie de o placă Basys3, un cablu USB pentru coonectarea acesteia la calculatorul pe care a fost generat bistream-ul ce urmează a fi încărcat pe placă, o tastatură și un calculator pe care să fie instalat mediul de dezvoltare Vivado 2019.2 Design Suite de la Xilinx, în care a fost scris codul proiectului, și anume: limbajul de programare VHDL (Very High Description Language) de descriere a hardware-ului.

Primul pas este realizarea conexiunilor dintre placă și tastatură și dintre placă și calculator cu ajutorul cablurilor. Mai apoi se deschide mediul de dezvoltare Vivavdo, se realizează sinteza, implementarea și scrierea bistream-ului. Pentru a fi posibilă scrierea bitstream-ului, pe lângă componentele descrise în limbajul VHDL, mai este nevoie și de fișierul de constrângeri specific plăcii folosite și specificate port-urile cu denmirile folosite în modulul principal, unde s-a făcut conexiunea între toate componentele, anozii și catozii plăcii, dar și între placă și tatatură. dar și Mai apoi se deschide meniul „Program and debug" și se selectează opțiunea „Open Hardware Manager" pentru a conecta placa la calculator și a încărca bistream-ul. Odată încărcat bitstream-ul se va putea testa programul prin apăsarea succesivă a unor taste și observarea comportamentului afișorului ca și consecința a acțiunii sale asupra tastelor.

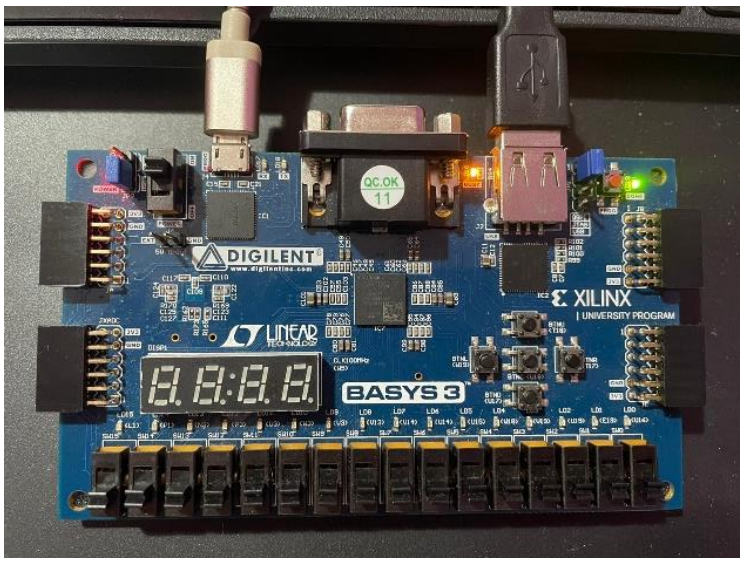


Figura 5.1. Apăsarea unei taste nerecunoscute

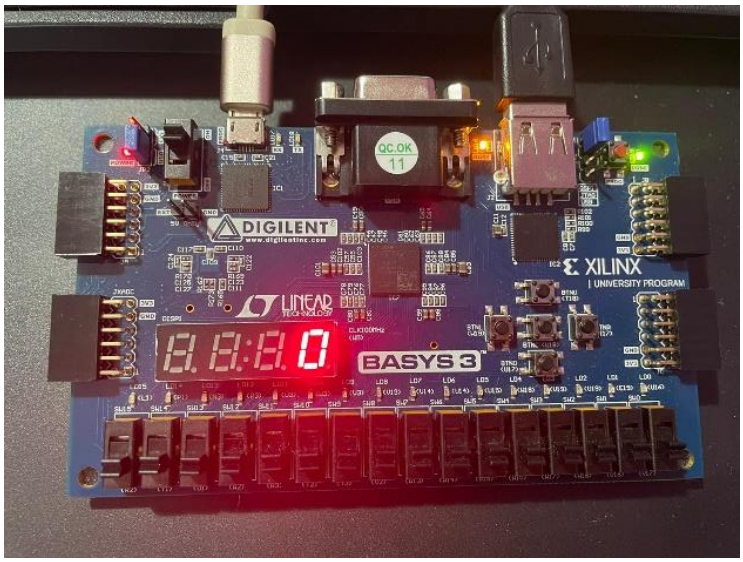


Figura 5.2. Apăsarea tastei corespunzătoare cifrei “0”

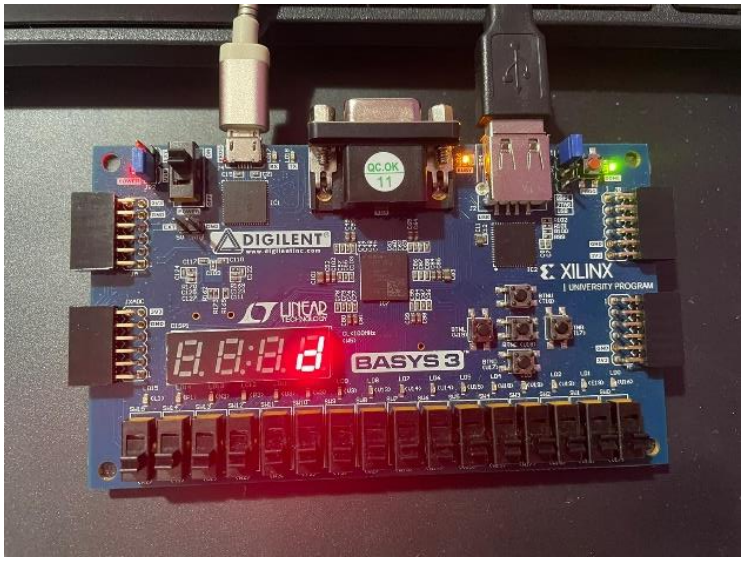


Figura 5.1. Apăsarea tastei corespunzătoare literei “d”

Pentru a vizualiza rezultatele pentru fiecare cifră și literă pe anodul plăcii Basys3 se poate accesa link-ul următor: https://www.youtube.com/watch?v=PD3SYBVecS4&feature=youtu.be

6. Concluzii

În urma acestui proiect s-a realizat conexiunea dintre placa Basys3 și o tastatură prin intermediul interfeței PS/2, făcându-se astfel posibilă afișarea caracterelor reprezentate de tastele apăsate ale tastaturii de către utilizator. Această problemă propusă de către acest proiect, a fost subdivizată în trei subprobleme, pentru fiecare dintre acestea concepându-se câte o componentă care să le rezolve, iar mai apoi, prin intermediul unui modul principal, s-a realizat conexiunea acestora pentru atingerea obiectivului final, cu ajutorul constrângerilor specifice plăcii. Cele trei module asignate celor trei subprobleme se ocupă de reazlizarea conexiunii între cele două componente hardware, filtrarea oscilațiilor tastelor tastaturii și afișarea caracterelor în funcție de codul receptat de la tastatură de către placă.

Avantajul soluției alese este modul foarte bine structurat în care s-a ales realizarea rezolvării problemei, prin intermediul unui număr succint de componente. Unul dintre dezavantajele evidente ale acestei implementări ar putea fi imposibilitatea afișării tuturor caracterelor din cazua limitării pe care le impune afișorul cu 7 segmente pe care-l pune la dispoziție placa aleasă pentru demonstrarea funcționalității proiectului.

Acest proiect ar putea fi folosit pentru dezvoltarea unor aplicații care necesită atât utilizarea unui limbaj software, cât și hardware. Realizarea unei componente care va transmite date dinspre placă înspre computer, pe linia de date „Tx", în limbajul de programare VHDL, va permite utilizarea împreună a tastaturii, a plăcii Basys3, a mediului de programare Vivado, a limbajului de programare VHDL, împreună cu un alt limbaj de programare, de data aceasta software și mediului său de programare. Ideea combinării tuturor mai sus menționate ar putea face posibilă afișarea anumitor date transmise de catre mediul software, pe placa Basys3, sau invers. Totodată, daca componenta hardware conectată ar fi alta, s-ar putea transmite date și către aceatsa, spre exemplu un mouse.

Ca și dezvoltări ulterioare ale implementării pe care o propun, ar fi afișarea caracterelor pe monitor prin intermediul unei aplicații dezvoltate într-un limbaj de programare software, spre exmplu Python. Pentru ca această dezvoltare să fie posibilă, nu ar fi de ajuns realizarea acestei aplicații, fiind nevoie de implementarea unei componente care să facă posibilă transmiterea datelor dinspre placă înspre aplicație. Totodată ar fi nevoie și de o decodificare a caracterelor din codul primit de la tastatură, în cod ASCII pentru a putea afișa caracterele corespunzător în aplicație.

Bibliografie

[1] https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual
[2] http://ace.upg-ploiesti.ro/cursuri/pla/curs_pla.pdf
[3] https://en.wikipedia.org/wiki/PS/2_port
[4] http://www.burtonsys.com/ps2_chapweske.htm

Anexe

Anexa A
main.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity modulul_principal is
   generic(
              frecv : integer := 100_000_000;
              keyboard_counter : integer := 9);
   port(
             -- semnale de intrare
             clk : in std_logic;
             start : in std_logic;
             rst : in std_logic;
             keyboard_clk : in std_logic;
             keyboard_data : in std_logic;
             -- semnalel de iesire
             an : out std_logic_vector (3 downto 0);
            seg : out std_logic_vector (7 downto 0));
end entity;

architecture behavioral of modulul_principal is

--componente
component keyboard_to_board is
 generic(
        frecv : integer := 100_000_000;
        counter : integer := 9);
 port(
        clk : in std_logic;
        keyboard_clk : in std_logic;
        keyboard_data : in std_logic;
        keyboard_code0 : out std_logic_vector(7 downto 0);
        keyboard_code1 : out std_logic);
end component;

component debouncer is
   port (
         clk : in std_logic;
         rst : in std_logic;
         din : in std_logic;
         qout : out std_logic);
```

```
end component;

component decodificator is
   port (
        -- semnalele de intrare
        code : in std_logic_vector(7 downto 0);
        --semnale de iesire
        an : out std_logic_vector (3 downto 0);
        seg : out std_logic_vector (7 downto 0));
end component;

--semnale
signal keyboard_code0: std_logic_vector(7 downto 0);
signal keyboard_code1: std_logic;

begin

kb_to_b: keyboard_to_board generic map(frecv, keyboard_counter) port map (clk, keyboard_clk,
keyboard_data, keyboard_code0, keyboard_code1);

display: decodificator port map (keyboard_code0, an, seg);

end behavioral;
```

Anexa B
decodificator.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity decodificator is
   port (
        -- semnalele de intrare
        code : in std_logic_vector(7 downto 0);
        --semnale de iesire
        an : out std_logic_vector (3 downto 0);
        seg : out std_logic_vector (7 downto 0));
end entity;

architecture behavioral of decodificator is
begin

process(code)
begin

   case code is
      when x"45" => seg <= "00000011"; -- 0
      when x"16" => seg <= "10011111"; -- 1
      when x"1e" => seg <= "00100101"; -- 2
      when x"26" => seg <= "00001101"; -- 3
      when x"25" => seg <= "10011001"; -- 4
      when x"2e" => seg <= "01001001"; -- 5
      when x"36" => seg <= "01000001"; -- 6
      when x"3d" => seg <= "00011111"; -- 7
      when x"3e" => seg <= "00000001"; -- 8
      when x"46" => seg <= "00001001"; -- 9
      when x"1C" => seg <= "00010001"; -- a
      when x"32" => seg <= "11000001"; -- b
      when x"21" => seg <= "01100011"; -- c
      when x"23" => seg <= "10000101"; -- d
      when x"24" => seg <= "01100001"; -- e
      when x"2b" => seg <= "01110001"; -- f
      when x"34" => seg <= "11111101"; -- g
      when x"33" => seg <= "11010001"; -- h
      when x"43" => seg <= "11011111"; -- i
      when x"3b" => seg <= "10000111"; -- j
      when x"42" => seg <= "11111101"; -- k
      when x"4b" => seg <= "11100011"; -- l
      when x"3a" => seg <= "11111101"; -- m
      when x"31" => seg <= "11010101"; -- n
      when x"44" => seg <= "11000101"; -- o
      when x"4d" => seg <= "00110001"; -- p
```

```
      when x"15" => seg <= "00011001"; -- q
      when x"2d" => seg <= "11110101"; -- r
      when x"1b" => seg <= "11111101"; -- s
      when x"2c" => seg <= "11111101"; -- t
      when x"3c" => seg <= "10000011"; -- u
      when x"2a" => seg <= "11000111"; -- v
      when x"1d" => seg <= "11111101"; -- w
      when x"22" => seg <= "11111101"; -- x
      when x"35" => seg <= "10001001"; -- y
      when x"1a" => seg <= "11111101"; -- z
      when others => seg <= "11111111";
   end case;
end process;

an <= "1110";

end behavioral;
```

Anexa C
debouncer.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity debouncer is
   generic (
         counter : integer := 20);
   port(
         -- semnale de intrare
         clk : in std_logic;
         din : in std_logic;
         -- semnalul de iesire
         qout : out std_logic);
end entity;

architecture Behavioral of debouncer is

signal counter_set : std_logic;
signal counter_out : std_logic_vector(counter downto 0) := (others =>'0');
signal bist : std_logic_vector(1 downto 0);

begin

 counter_set <= bist(0) xor bist(1);

process(clk)
begin

 if( rising_edge(clk) ) then
   bist(0) <= din;
   bist(1) <= bist(0);

   if(counter_set = '1') then
      counter_out <= (others => '0');
   elsif(counter_out(counter) = '0') then
      counter_out <= counter_out + 1;
   else
      qout <= bist(1);
   end if;
 end if;

 end process;

end Behavioral;
```

Anexa D
Keyboard_to_board.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity keyboard_to_board is
 generic(
        frecv : integer := 100_000_000;
        counter : integer := 9);
 port(
        --semnale de intrare
        clk : in std_logic;
        keyboard_clk : in std_logic;
        keyboard_data : in std_logic;
        --semnale de iesire
        keyboard_code0 : out std_logic_vector(7 downto 0);
        keyboard_code1 : out std_logic);
end entity;

architecture Behavioral of keyboard_to_board is

 --componente
component debouncer is
   generic (
          counter_size : integer := 20);
   port(
          -- semnale de intrare
          clk : in std_logic;
          button : in std_logic;
          -- semnalul de iesire
          result : out std_logic);
end component;

 --semnale
 signal dc_out : std_logic_vector(1 downto 0);
 signal keyboard_clk_out : std_logic;
 signal keyboard_data_out : std_logic;
 signal stored_data : std_logic_vector(10 downto 0);
 signal err : std_logic;
 signal counter0 : integer range 0 to frecv / 18_000;

begin

 process(clk)
 begin
```

```
    if( rising_edge(clk) ) then
      dc_out(0) <= keyboard_clk;
      dc_out(1) <= keyboard_data;
    end if;

 end process;

 debounce_ps2_clk: debouncer generic map(counter) port map(clk, dc_out(0), keyboard_clk_out);
 debounce_ps2_data: debouncer generic map(counter) port map(clk, dc_out(1), keyboard_data_out);

 process(keyboard_clk_out)
 begin

   if( falling_edge(keyboard_clk_out) ) then
      stored_data(10) <= keyboard_data_out;
      stored_data(9 downto 0) <= stored_data(10 downto 1);
   end if;

 end process;

process(stored_data)
begin

   err <= not (not stored_data(0) and stored_data(10) and (stored_data(1)xor
        stored_data(2) xor stored_data(3) xor stored_data(4) xor stored_data(5) xor
        stored_data(6) xor stored_data(7) xor stored_data(8) xor stored_data(9)));

end process;

process(clk)
begin

 if( rising_edge(clk) )then
   if(keyboard_clk_out = '0') then
      counter0 <= 0;
   elsif(counter0 /= frecv/18_000) then
      counter0 <= counter0 + 1;
   end if;

   if( counter0= frecv / 18_000 and err = '0' ) then
      keyboard_code1<= '1';
      keyboard_code0 <= stored_data(8 downto 1);
   else
      keyboard_code1 <= '0';
   end if;
 end if;

end process;

end Behavioral;
```

Anexa E
constraints.xdc

```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
	set_property IOSTANDARD LVCMOS33 [get_ports clk]
	create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
#set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
#set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
#set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
#set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
#set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
#set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
#set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
#set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
#set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
#set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
#set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
#set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
#set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
#set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
#set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
#set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]
```

```
## LEDs
#set_property PACKAGE_PIN U16 [get_ports {led[0]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
#set_property PACKAGE_PIN E19 [get_ports {led[1]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
#set_property PACKAGE_PIN U19 [get_ports {led[2]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
#set_property PACKAGE_PIN V19 [get_ports {led[3]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
#set_property PACKAGE_PIN W18 [get_ports {led[4]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
#set_property PACKAGE_PIN U15 [get_ports {led[5]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
#set_property PACKAGE_PIN U14 [get_ports {led[6]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
#set_property PACKAGE_PIN V14 [get_ports {led[7]}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
#set_property PACKAGE_PIN V13 [get_ports {led[8]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
#set_property PACKAGE_PIN V3 [get_ports {led[9]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
#set_property PACKAGE_PIN W3 [get_ports {led[10]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
#set_property PACKAGE_PIN U3 [get_ports {led[11]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
#set_property PACKAGE_PIN P3 [get_ports {led[12]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
#set_property PACKAGE_PIN N3 [get_ports {led[13]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
#set_property PACKAGE_PIN P1 [get_ports {led[14]}]
	#set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
#set_property PACKAGE_PIN L1 [get_ports {ps2_code_new}]
#	set_property IOSTANDARD LVCMOS33 [get_ports {ps2_code_new}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {seg[7]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {seg[7]}]
set_property PACKAGE_PIN W6 [get_ports {seg[6]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property PACKAGE_PIN U8 [get_ports {seg[5]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN V8 [get_ports {seg[4]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property PACKAGE_PIN U5 [get_ports {seg[3]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property PACKAGE_PIN V5 [get_ports {seg[2]}]
	set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property PACKAGE_PIN U7 [get_ports {seg[1]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]

set_property PACKAGE_PIN V7 [get_ports seg[0]]
        set_property IOSTANDARD LVCMOS33 [get_ports seg[0]]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]


##Buttons
#set_property PACKAGE_PIN U18 [get_ports btnC]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnC]
#set_property PACKAGE_PIN T18 [get_ports btnU]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnU]
#set_property PACKAGE_PIN W19 [get_ports btnL]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnL]
#set_property PACKAGE_PIN T17 [get_ports btnR]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnR]
#set_property PACKAGE_PIN U17 [get_ports btnD]
        #set_property IOSTANDARD LVCMOS33 [get_ports btnD]



##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
```

```
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]


##Pmod Header JB
##Sch name = JB1
#set_property PACKAGE_PIN A14 [get_ports {JB[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[0]}]
##Sch name = JB2
#set_property PACKAGE_PIN A16 [get_ports {JB[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[1]}]
##Sch name = JB3
#set_property PACKAGE_PIN B15 [get_ports {JB[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[2]}]
##Sch name = JB4
#set_property PACKAGE_PIN B16 [get_ports {JB[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[3]}]
##Sch name = JB7
#set_property PACKAGE_PIN A15 [get_ports {JB[4]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[4]}]
##Sch name = JB8
#set_property PACKAGE_PIN A17 [get_ports {JB[5]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[5]}]
##Sch name = JB9
#set_property PACKAGE_PIN C15 [get_ports {JB[6]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[6]}]
##Sch name = JB10
#set_property PACKAGE_PIN C16 [get_ports {JB[7]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JB[7]}]


##Pmod Header JC
##Sch name = JC1
#set_property PACKAGE_PIN K17 [get_ports {JC[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[0]}]
##Sch name = JC2
#set_property PACKAGE_PIN M18 [get_ports {JC[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[1]}]
##Sch name = JC3
#set_property PACKAGE_PIN N17 [get_ports {JC[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[2]}]
##Sch name = JC4
#set_property PACKAGE_PIN P18 [get_ports {JC[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {JC[3]}]
##Sch name = JC7
#set_property PACKAGE_PIN L17 [get_ports {JC[4]}]
```

#set_property IOSTANDARD LVCMOS33 [get_ports {JC[4]}]
##Sch name = JC8
#set_property PACKAGE_PIN M19 [get_ports {JC[5]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JC[5]}]
##Sch name = JC9
#set_property PACKAGE_PIN P17 [get_ports {JC[6]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JC[6]}]
##Sch name = JC10
#set_property PACKAGE_PIN R18 [get_ports {JC[7]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JC[7]}]

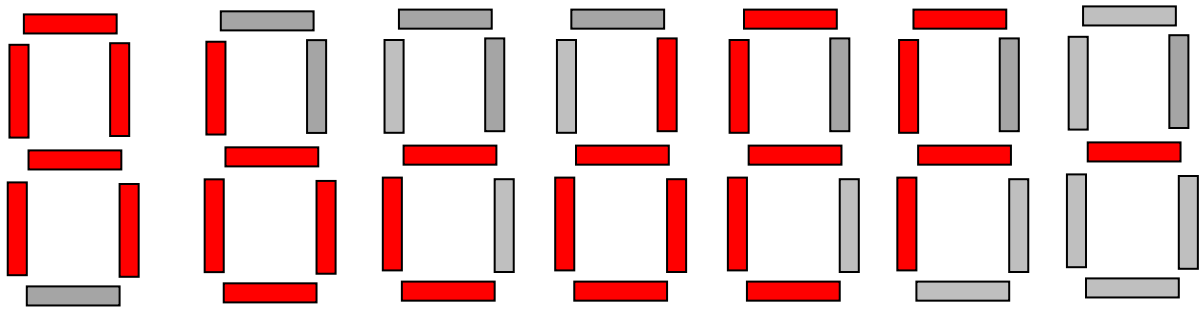
##Pmod Header JXADC
##Sch name = XA1_P
#set_property PACKAGE_PIN J3 [get_ports {JXADC[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[0]}]
##Sch name = XA2_P
#set_property PACKAGE_PIN L3 [get_ports {JXADC[1]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[1]}]
##Sch name = XA3_P
#set_property PACKAGE_PIN M2 [get_ports {JXADC[2]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[2]}]
##Sch name = XA4_P
#set_property PACKAGE_PIN N2 [get_ports {JXADC[3]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[3]}]
##Sch name = XA1_N
#set_property PACKAGE_PIN K3 [get_ports {JXADC[4]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[4]}]
##Sch name = XA2_N
#set_property PACKAGE_PIN M3 [get_ports {JXADC[5]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[5]}]
##Sch name = XA3_N
#set_property PACKAGE_PIN M1 [get_ports {JXADC[6]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[6]}]
##Sch name = XA4_N
#set_property PACKAGE_PIN N1 [get_ports {JXADC[7]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {JXADC[7]}]

##VGA Connector
#set_property PACKAGE_PIN G19 [get_ports {vgaRed[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[0]}]
#set_property PACKAGE_PIN H19 [get_ports {vgaRed[1]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[1]}]
#set_property PACKAGE_PIN J19 [get_ports {vgaRed[2]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[2]}]
#set_property PACKAGE_PIN N19 [get_ports {vgaRed[3]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {vgaRed[3]}]
#set_property PACKAGE_PIN N18 [get_ports {vgaBlue[0]}]

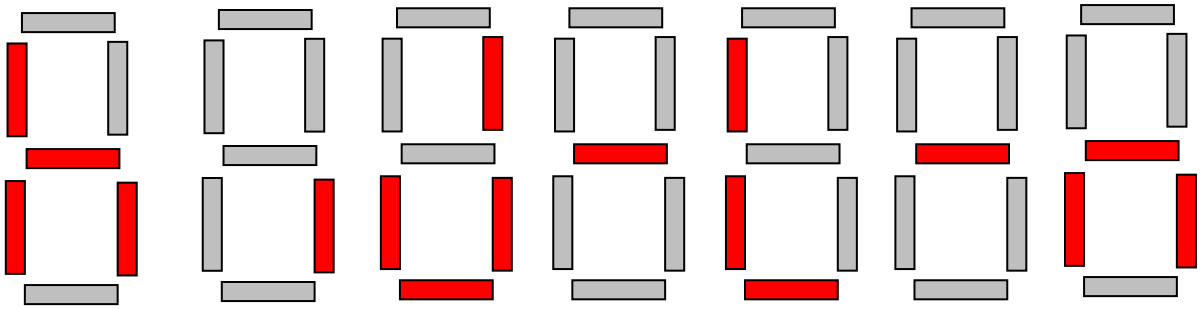
```
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[0]}]
#set_property PACKAGE_PIN L18 [get_ports {vgaBlue[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[1]}]
#set_property PACKAGE_PIN K18 [get_ports {vgaBlue[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[2]}]
#set_property PACKAGE_PIN J18 [get_ports {vgaBlue[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaBlue[3]}]
#set_property PACKAGE_PIN J17 [get_ports {vgaGreen[0]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[0]}]
#set_property PACKAGE_PIN H17 [get_ports {vgaGreen[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[1]}]
#set_property PACKAGE_PIN G17 [get_ports {vgaGreen[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[2]}]
#set_property PACKAGE_PIN D17 [get_ports {vgaGreen[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {vgaGreen[3]}]
#set_property PACKAGE_PIN P19 [get_ports Hsync]
        #set_property IOSTANDARD LVCMOS33 [get_ports Hsync]
#set_property PACKAGE_PIN R19 [get_ports Vsync]
        #set_property IOSTANDARD LVCMOS33 [get_ports Vsync]


##USB-RS232 Interface
#set_property PACKAGE_PIN B18 [get_ports RsRx]
  #set_property IOSTANDARD LVCMOS33 [get_ports RsRx]
#set_property PACKAGE_PIN A18 [get_ports RsTx]
        #set_property IOSTANDARD LVCMOS33 [get_ports RsTx]


##USB HID (PS/2)
set_property PACKAGE_PIN C17 [get_ports keyboard_clk]
        set_property IOSTANDARD LVCMOS33 [get_ports keyboard_clk]
        set_property PULLUP true [get_ports keyboard_clk]
set_property PACKAGE_PIN B17 [get_ports keyboard_data]
        set_property IOSTANDARD LVCMOS33 [get_ports keyboard_data]
        set_property PULLUP true [get_ports keyboard_data]


##Quad SPI Flash
##Note that CCLK_0 cannot be placed in 7 series devices. You can access it using the
##STARTUPE2 primitive.
#set_property PACKAGE_PIN D18 [get_ports {QspiDB[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[0]}]
#set_property PACKAGE_PIN D19 [get_ports {QspiDB[1]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[1]}]
#set_property PACKAGE_PIN G18 [get_ports {QspiDB[2]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[2]}]
#set_property PACKAGE_PIN F18 [get_ports {QspiDB[3]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {QspiDB[3]}]
#set_property PACKAGE_PIN K19 [get_ports QspiCSn]
#set_property IOSTANDARD LVCMOS33 [get_ports QspiCSn]
```

Anexa F
Reprezentarea caracterelor pe afișorul 7 segmente

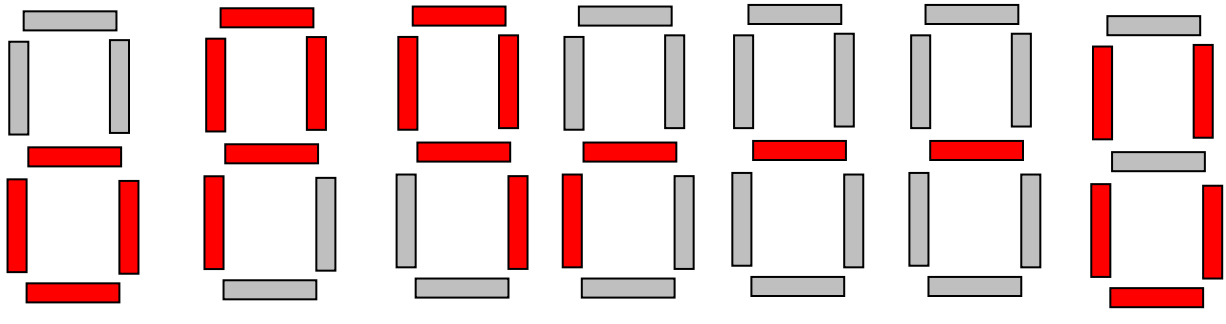
LITERE



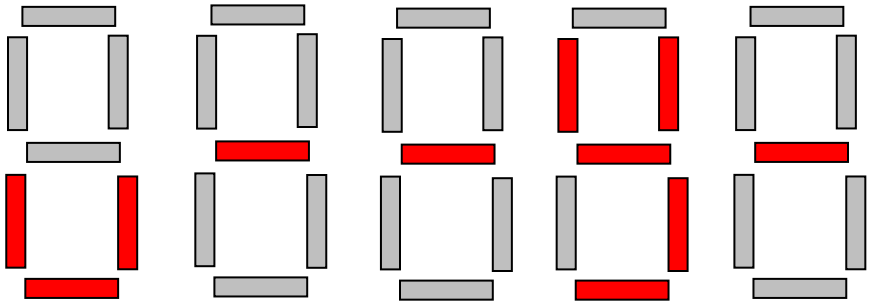
A B C D E F G



H I J K L M N

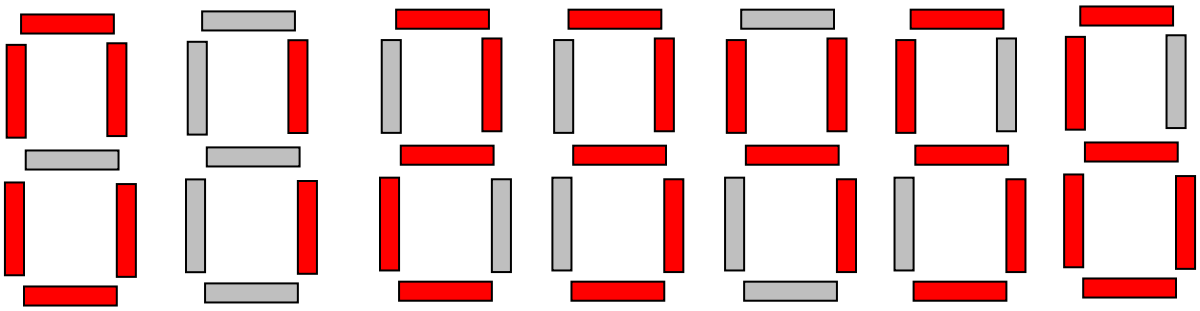


O P Q R S T U

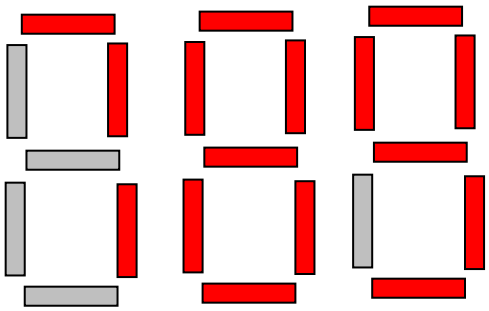


V W X Y Z

CIFRE



0 1 2 3 4 5 6



7 8 9