

**Facultate: Automatică și Calculatoare**

**Specializare: Calculatoare și Tehnologia Informației**

**Disciplina: Proiectarea Sistemelor Numerice**

**Proiect: Controller de Tastatura PS-2**

Studenti : Chiciudean Vivian

Herman Felician-Nicu

Seria: B

Grupa: 30219

**Cuprins**

1.Specificație

* Cerință
* Despre proiect
* Protocolul tastaturii PS2
* Tastatură
* Placa de dezvoltare Basys 2
* Limbajul VHDL si mediul de dezvoltare Xilinx ISE

2.Schema bloc cu componentele principale + schema detaliată

3.Lista de componente. Proiectarea şi implementarea

* Scheme bloc
* Proiectarea componentelor
* Semnificaţia notaţiilor
* Schema interna

4.Justificarea soluţiei alese

5.Utilizare şi rezultate

* Resurse utilizate
* Paşi necesari pentru utilizare
* Rezultate

6.Posibilităţi de dezvoltare ulterioară

**Specificație**

***Cerință:***

**A.12.** Să se proiecteze un controller de tastatură PS2. Se cere citirea tastelor și afișarea de caractere corespunzătoare pe afișorul cu 7 segmente. Se vor vedea ultimele 4 simboluri, iar tastele de control vor avea roluri speciale (de exemplu: Enter începe un rând nou, ștergând afișorul). Poziția punctului pe afișor va fi controlată cu ajutorul săgeților.

***Despre proiect:***

Comunicarea dintre utilizator și sistem este un element fundamental în domeniul calculatoarelor, iar dacă această comunicare nu ar exista, sistemul nu ar avea cum să primească și să ințeleagă datele de intrare. Pentru a face posibilă această comunicare directă cu utilizatorul sistemele sunt dotate cu porturi de intrare folosite pentru a conecta diferite module periferice. Una dintre cele mai utilizate periferice, fără de care manipularea sistemelor de calcul ar fi imposibilă este tastatura. Tastatura este perifericul care prin folosirea butoanelor marcate cu litere, cifre si diverse alte taste funcționale permite introducerea de date dintr-un limbaj folosit de om, într-un limbaj înțeles de sistem, creându-se astfel un fel de interfață cu utilizatorul.

Proiectul de față își propune implementarea unui controller pentru interfața PS2 a tastaturii, a cărui rezultat este afișarea corespunzătoare a datelor de ieșire. Pentru fiecare tastă apăsată de la tastatură se va afișa pe afișorul cu 7 segmente acel caracter. În cadrul proiectului s-a folosit: mediul de dezvoltare Xilinx ISE și limbajul de descriere hardware VHDL, o placă de dezvoltare Basys2 și o tastatură standard PS2 cu 6 pini.

Tastatura face conexiunea la calculator printr-o tehnologie denumită PS-2. Scopul principal pe care vrem să îl atingem este cel de a crea acea conexiune PS-2 cu ajutorul plăcii DIGILENT Basys2. Dorim ca la momentul apăsarii unui buton de pe tastatură, aceasta să fie recunoscut si transmis plăcii iar pe afișajul acesteia să apară caracterul corespunzător. Protocolul folosit ințelege caracterele date sub forma unui cod in hexazecimal transmis de la perific. Rezultatul va fi afisat in mod instant din cauza vitezei placii DIGILENT Basys2 și a metodei prin care realizam transmisia și procesarea de date. Viteza este ridicată deoarece folosim stări pentru a descrie procesul de transmisie și procesare.

***Protocolul tastaturii PS2:***

Literatura de specialitate care acoperă domeniul calculatoarelor și implicit subdomeniul perifericelor și sistemelor de intrare-ieșire care folosesc protocolul PS2 este destul de restrânsă limitându-se la cursuri de specialitate și diferite publicații sau specificații ale producătorilor. Protocolul PS2 prevede comunicarea tastaturii utilizând două tipuri de bază de conectori 5-pin DIN și 6-pin DIN. Ambele funcționează în mod electric și conversia între cele două, dacă situația o cere, este foarte ușor de realizat folosindu-se un adaptor care de asemenea respectă o construcție standard. În literatura de specialitate perifericele care folosesc conectorul 5-pin sunt cunoscute sub denumirea de periferice AT, în timp ce acelea care folosesc conectorul 6-pin sunt denumite pur și simplu periferice PS2.

Protocolul PS2 presupune comunicare bidirecțională între două sisteme, astfel avem comunicare gazdă-periferic și comunicare periferic-gazdă ambele realizându-se după anumite reguli standard. Comunicarea se face conectând cei doi conectori male si female, iar transmisia sau recepția se poate face utilizând cele două căi: una de date și una de clock, din cele 6 în cazul 6-pin DIN, respectiv 5 în cazul 5-pin DIN.

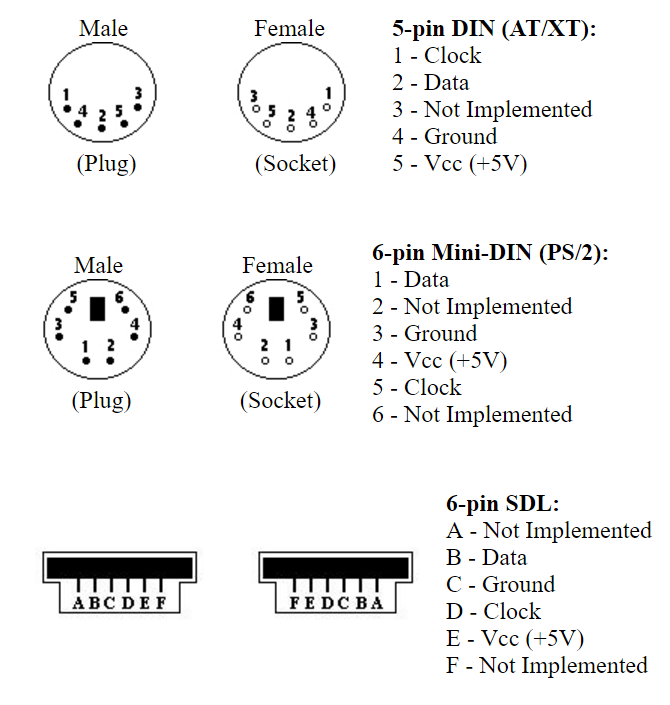
Pentru proiectul de față am folosit protocolul PS2, 6-pin Mini-DIN (PS2) care folosește pentru alimentare pinul 3 – Ground si pinul 4 –Vcc, iar pentru clock si date pinul 5, respectiv pinul 1. Când ambele linii de comunicații sunt *high* conexiunea se află în starea *idle*. În aceasta situație tastatura are permisiunea de a începe transmiterea datelor. Comunicația este inhibată când linia de tact este ținută pe l*ow*, iar când aceasta este ținută *high* și linia de date este ținută *low*, calculatorul generează o cerere de transmitere a datelor către tastatură. Semnalul de tact al tastaturii este cuprins între 10 si 16.7 kHz și este esențial ca unitatea de control să țină cont de această frecvență în transmiterea/recepția datelor. Sincronizarea dintre generatorul de tact al calculatorului și cel al tastaturii este importană pentru o bună comunicare. Pentru a putea primi date de la tastatură, dispozitivul trebuie să țină linia de tact high pentru cel putin 30-50 microsecunde, iar apoi low pentru alte 30-50 microsecunde.

Tastatura generează întotdeauna semnalul de ceas, dar gazda are întotdeauna un control final asupra comunicării. Linia de ceas trebuie să fie în mod continuu high pentru cel puțin 50 de microsecunde, înainte ca dispozitivul să poată să înceapă să transmită datele sale.

*Data = high, Clock = high: Starea inactiva.*

*Data = high, Clock = low: Comunicare inhibata.*

*Data = low, Clock = high: Cerere de trimitere a gazdei*

Un pachet trimis de către tastatură către dispozitivul de control conține 11 biti ( figura de mai jos):

1 bit de start (care este tot timpul 0);

8 biți de date (primul bit este cel mai puțin semnificativ);

1 bit de paritate;

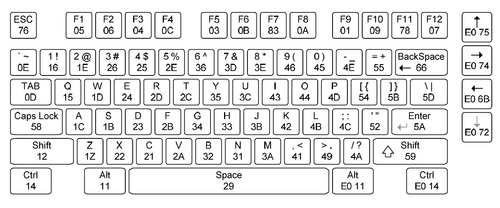
1 bit de stop (care este tot timpul 1);

Tastatura scrie un bit pe linia de date atunci când linia de tact este high și este citită de către gazdă, atunci când linia de tact este low:



Biții de start și stop au rolul de a semnaliza începutul transmiterii de date, respectiv sfarșitul transmisiei. Bitul de paritate are rolul de verificare a corectitudiniii datelor din pachetul curent și este setat la 1 în cazul în care în pachet se află un numar par de biți high, sau la 0 în cazul în care în pachet există un număr impar de biți high.

Modul în care calculatorul interpretează codul primit ține de felul în care acesta a fost programat. Majoritatea calculatoarelor lasă această sarcină plăcii de bază, care transformă codul primit de la tastatură într-un cod care poate fi interpretat. Valorile codurilor se găsesc în imaginea de mai jos.



***Tastatura:***

Tastatura este o componentă hardware perifericǎ a calculatorului ce permite utilizatorului sǎ introducǎ în unitatea centralǎ a acestuia date (litere, cifre și semne speciale) prin apǎsarea unor taste. Cele mai folosite tastaturi sunt cele de tip QWERTY. Fiecare tastă are asociatǎ un numǎr de identificare care poartǎ denumirea de "cod de scanare". La apǎsarea unei taste, tastatura trimite sistemului de calcul codul de scanare corespunzător tastei respective (un număr intreg de la 1 la „n” - numǎrul de taste). La primirea codului de scanare de la tastaturǎ, calculatorul face conversia între numǎrul primit și codul ASCII corespunzator, în logica binarǎ.

Tastatura folositǎ pentru proiect este una standard de tip PS2 care are funcționalitǎțile și butoanele de bazǎ, minime întrebuințǎrii la care este supusǎ. De asemenea de menționat este faptul că nu se folosesc tastele speciale de tip Alt, Control, butoanele de funcții F, Shift sau alte taste de acest gen, însă se folosesc săgețile, ENTER-ul și Caps Lock-ul pentru afișarea literelor mari.

***Placa de dezvoltare Basys 2:***

Placa de dezvoltare folosită pentru proiect este una dintre familiile FPGA, denumită BASYS 2, această placă fiind produsă de firma Digilent, ea oferind toate instrumentele necesare bunei functionări a aplicației. Placa Digilent Basys 2 este o platfomă de dezvoltare pentru circuite integrate.Basys 2 oferă vectori externi de memorie de dimensiuni mari, o colecție de dispositive de I/O folositoare și numeroase porturi, devenind astfel platforma ideală pentru experimentele cu sisteme digitale bazate pe FPGA. Placa este potrivită atât pentru circuitele logice simple cât și pentru sistemele digitale complexe. Toate semnalele externe sunt fără încărcare electrostatică și au protecție pentru scurt‐circuit, asigurând astfel o funcționare durabilă în orice mediu. Placa Basys o platformă ieftină, robustă și ușor de folosit de către oricine dorește să experimenteze dispozitivile FPGA și metodele moderne de design. Este centrată pe Spartan-3E FPGA și suportă toate circuitele necesare astfel încât design‐urile să fie implementate și folosite repede.

Printre componentele de care dispune putem enumera câteva care sunt de folos în realizarea conexiunii cu tastatura:

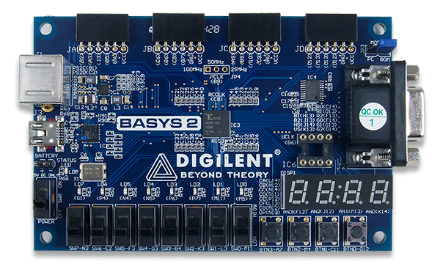
• portul de conectare PS2;

• patru afișoare de tip 7 segmente;

• opt întrerupǎtoare cu slide;

• opt LED–uri;

• patru butoane cu apǎsare temporarǎ;

****

Placa Basys include numeroase dispozitive pentru intrări și ieșiri, precum și numeroase porturi pentru date, permițând astfel implementarea diferitelor design‐uri fără a fi nevoie de alte componente.

***Intrări***: Comutatoare și butoane

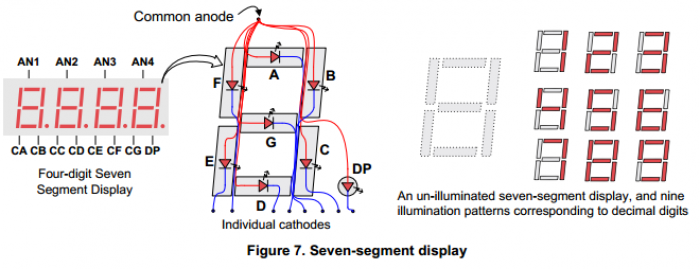
Patru butoane și opt comutatoare sunt oferite pentru intrările circuitului. Butoanele sunt în mod normal inactive și se activează în momentul în care sunt apăsate. Comutatoarele generează constant intrări reduse sau înalte în funcție de poziția lor. Butoanele și comutatoarele folosesc un rezistor serie pentru protecție împotriva scurt‐circuitelor (un scurt‐circuit ar putea avea loc dacă un pin FPGA asignat unui buton sau comutator ar fi definit în mod neintenționat ca un output).

***Ieșiri:*** LED‐uri

Opt LED‐uri sunt disponibile pentru ieșirile circuitului. Anodurile LED‐urilor sunt conduse din FPGA prin rezistori de 390 ohm, așa că un ‘1’ logic va lumina cu 3‐4 ms din unitatea curentă. Un al nouălea LED are rolul de LED care semnalizează pornirea și un al zecelea LED indică starea de programare a FPGA‐ului.

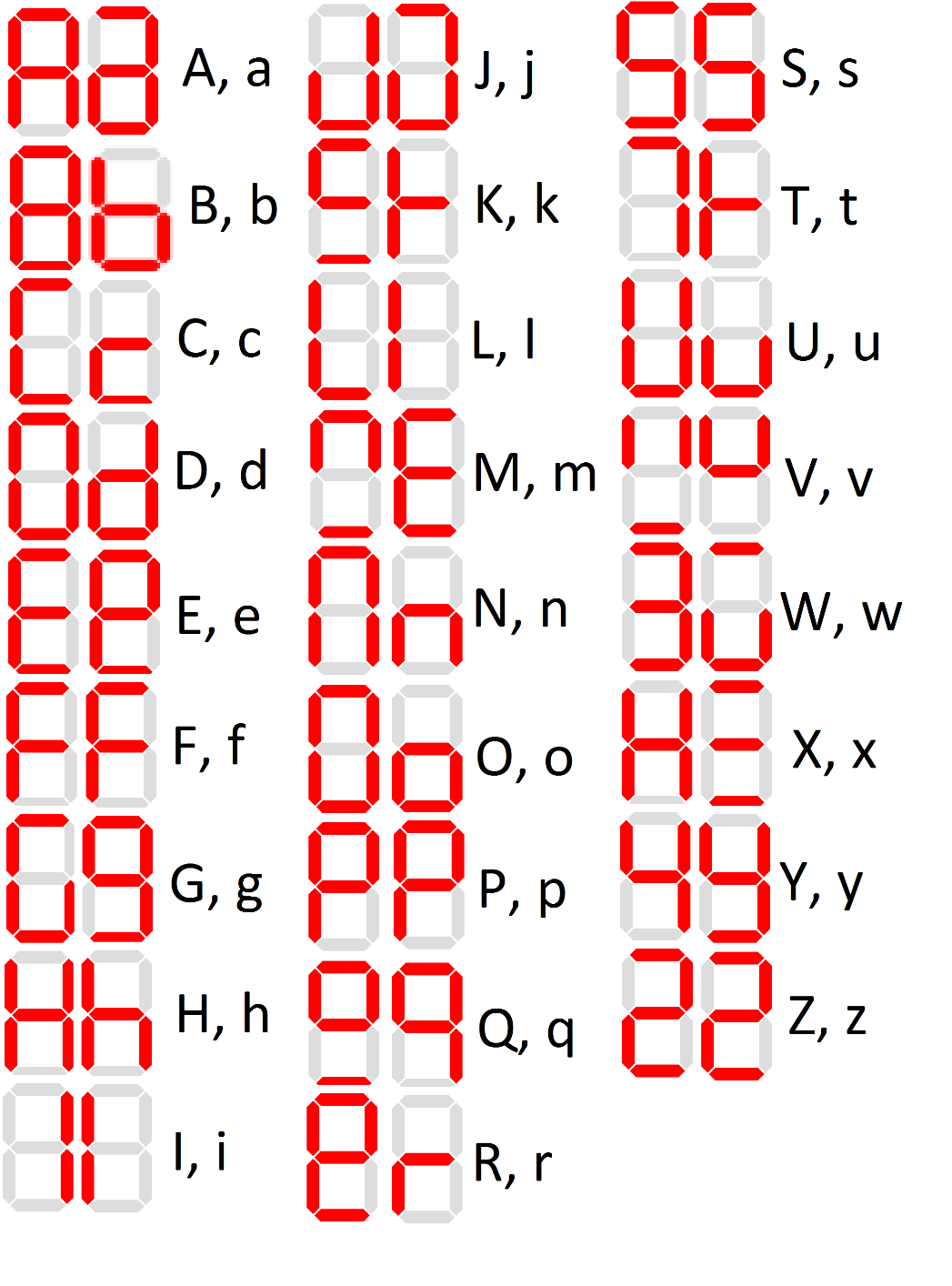
***Ieșiri:*** afișaj 7 – segmente

Placa Basys 2 conține un anod comun din patru cifre cu afișaj LED format din șapte segmente. Fiecare dintre cele patru cifre este compusă din șapte segmente sub forma cifrei 8, cu câte un LED integrat în fiecare segment. LED-urile segmentelor pot fi iluminate individual, așa că oricare dintre cele 128 de modele poate fi afișat pe o cifră prin iluminarea anumitor segmente și neiluminarea celorlalte. Dintre cele 128 de modele, cele mai folosite sunt cele zece corespunzătoare cifrelor zecimale. Anodurile celor șapte LED-uri care formează fiecare cifră sunt legate într-un anod comun, dar catodurile LED-urilor rămân separate. Semnalele anodului comun sunt disponibile ca 4 semnale de activare cifră pentru afișajul de tip 4-cifre.



Un circuit controller pentru scanarea afișajului poate fi folosit pentru a arăta un număr format din patru cifre pe acest afișaj. Acest circuit conduce semnalele de anod și modelele catozilor corespunzători fiecărei cifre într‐o succesiune continuă, repetitivă, la o rată de actualizare care este mai rapidă decât poate răspunde ochiul uman. Fiecare cifră este iluminată doar o pătrime din timp, dar pentru că ochiul uman nu poate percepe întunecarea cifrei până atunci când este iluminată din nou, cifra apare ca fiind iluminată continuu. Dacă rata de actualizare este micșorată până la un anumit punct (în jur de 45 Hz), atunci majoritatea persoanelor vor vedea display‐ul 8 pâlpâind. Pentru ca cele patru cifre să apară strălucitoare și iluminate continuu, este necesar ca toate patru să fie actualizate la 1‐16 ms.

În cadrul proiectului noi am ales diferite moduri de afișare a literelor speciale cum ar fi X, Y, Z,etc. În tabelul următor prezentăm concret în ce fel se afișează fiecare literă în parte.



Conectori 6‐pini

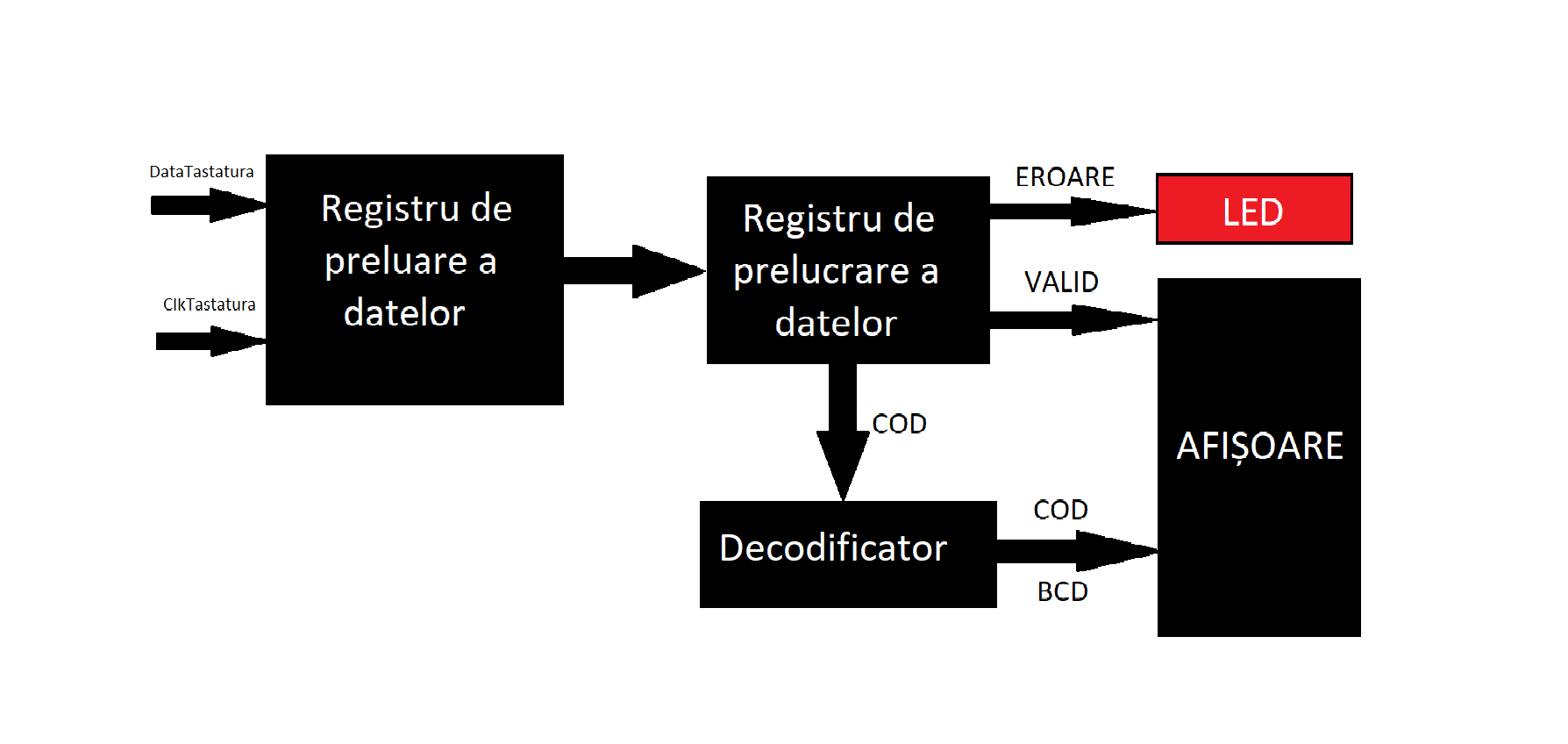
Placa Basys oferă un modul periferic de patru conectori 6‐pini. Fiecare conector este caracterizat prin Vcc, GND și patru semnale FPGAS unice. Toate cele patru circuite 6‐pin au rezistori pentru protecție împotriva scurt‐circuitelor și diode pentru protecție împotriva descărcărilor electrostatice. Diferite module de plăci 6‐pin oferite de către Digilent pot fi atașate acestui conector.

***Limbajul VHDL și mediul de dezvoltare Xilinx ISE***

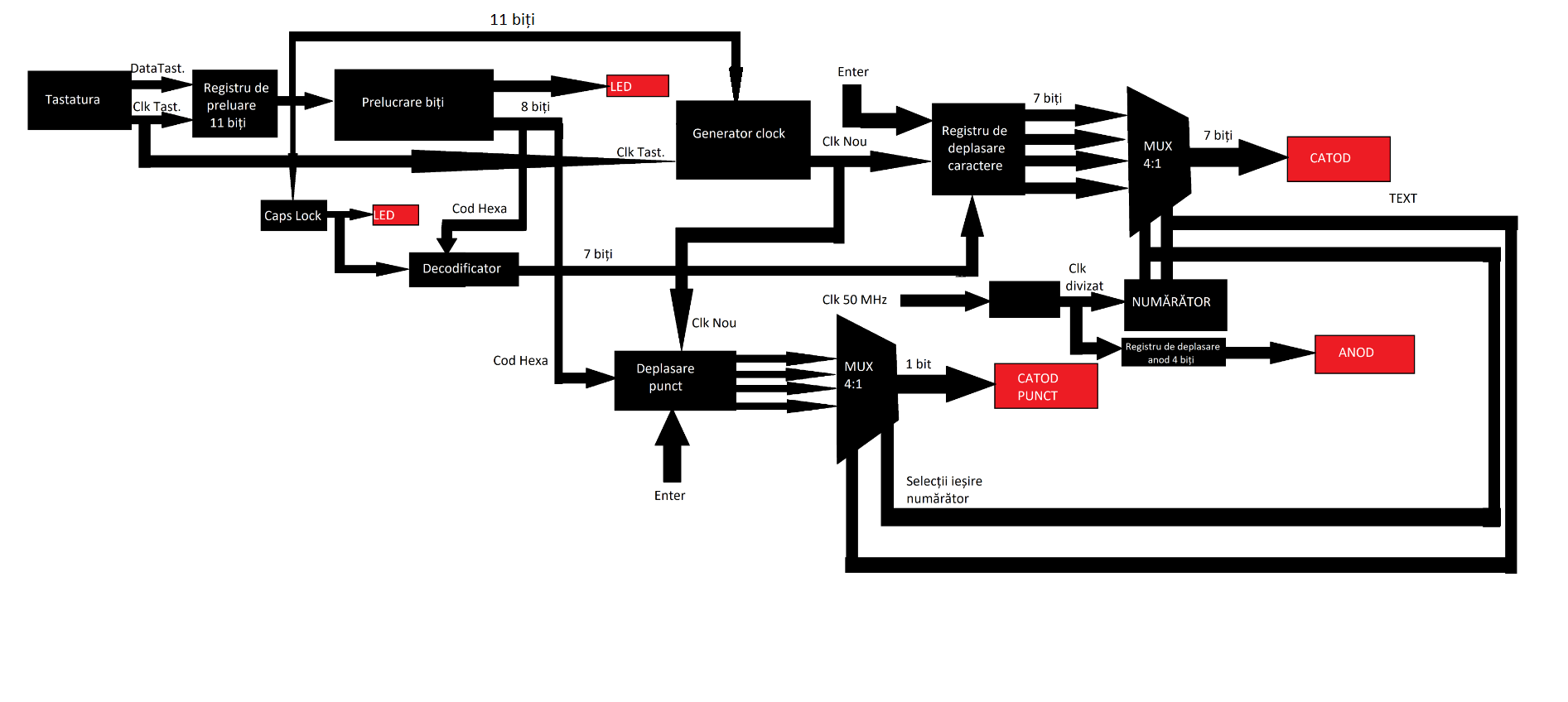
Unul dintre cele mai utilizate medii de dezvoltare a aplicațiilor care necesită descriere hardware este cel pus la dispoziție de firma Xilinx, denumit Xilinx ISE. În acest mediu s-a implementat și testat proiectul care implică folosirea protocolului PS2 cu tastatura. Întregul cod sursă a fost scris cu ajutorul editorului schematic şi al simulatorului din mediul de proiectare ACTIVE-HDL. ACTIVE-HDL, produs al firmei ALDEC, care este un mediu integrat pentru dezvoltarea de proiecte schematice sau bazate pe limbaje de descriere hardware: VHDL, Verilog, EDIF sau mixte. Limbajul folosit este VHDL unul dintre cele mai utilizate limbaje pentru proiecte care implicǎ descriere hardware. Testarea s-a efectuat folosind modulul de simulare pus la dispoziție tot de utilitarul Xilinx ISE.

**Schema bloc cu componentele principale**

Proiectul se poate împărți în trei mari componente. Prima dintre acestea este Registrul de preluare a datelor, care preia datele de la tastatura PS2 în mod secvențial. Cea de-a doua este componenta care se ocupă cu prelucrarea biților, care în cazul primirii unui cod corect, va activa cea de-a treia componentă, afisarea. Afișarea se ocupă de reprezentarea pe 7 segmente a codurilor transmise de la tastatură. În cazul detectării unei erori la codul primit, se va genera o eroare și se va aprinde un led de pe plăcuță.

****

**Schema detaliata**

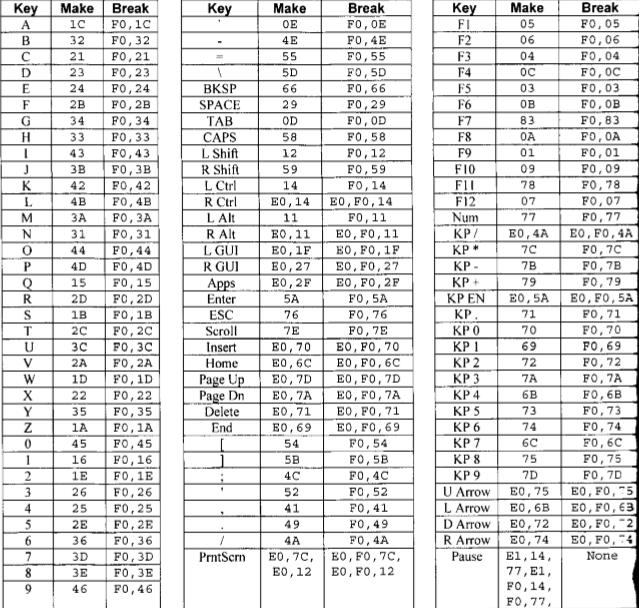
****

**Lista de componente. Proiectarea si implementarea**

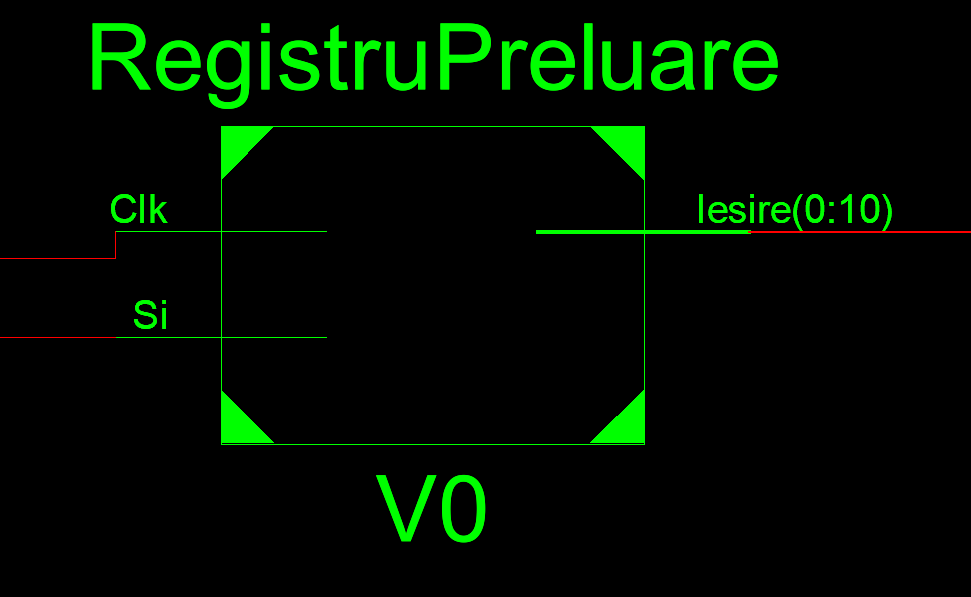
**Registru preluare biti:**

Proiectare componentă:

Înainte de toate, avem nevoie de un registru în care să preluăm datele provenite de la tastatură. Fiecare tastă are un cod standard. Acest cod se transmite secvenţial registrului de preluare. La apăsarea fiecărei taste se transmite un „Make code” (de exemplu pentru A: 1C), respectiv un „Break code” (pentru A: F0 1C). În tabelul următor vom ilustra codul fiecărei taste în parte:



Schema bloc:

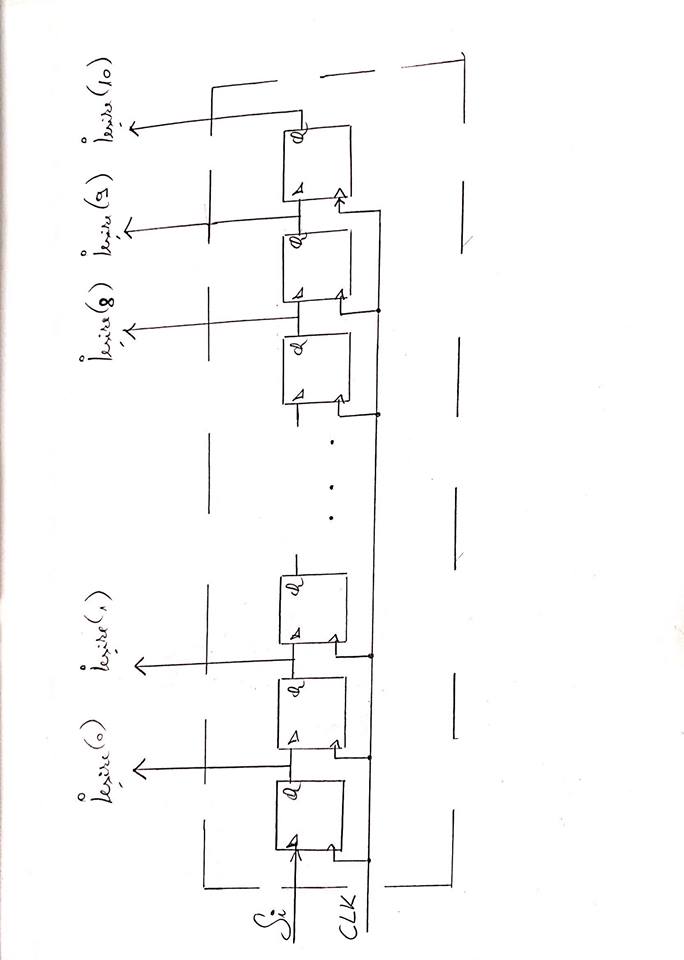


Semnificaţia notaţiilor:

• Clk: clock-ul de la tastatură

• Si: serial input, biții sunt introduși in mod secvențial

• Iesire: ieșirea registrului de preluare pe 11 biți; primul bit reprezintă bitul de start, următorii 8 codul standard al fiecărei taste, al 9-lea este bitul de paritate, iar bitul 10 este bitul de stop.

****

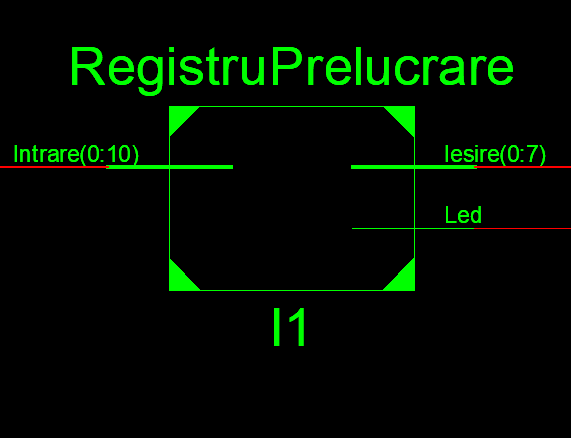
**Interiorul componentei este format din 11 bistabile de tip D, cu CLK activ pe frontul descrescator, comportandu-se ca si un registru de shiftare de tip serial-in, parallel-out.**

**Prelucrare biţi:**

Proiectare componentă:

În procesul transmiterii informaţiilor numerice, pot apărea erori. O metodă simplă de detectare a acestora constă în verificarea corectitudinii bitului de paritate, respectiv prin verificarea bitului de start şi de stop, care întotdeauna trebuie să aibă valoarea ‚0’, respectiv ‚1’. Corectitudinea bitului de paritate se verifică prin calcularea separată a acestuia, cu ajutorul unei porţi sau-exclusiv între biţii 1-8. La final, comparăm rezultatul acestui calcul cu bitul 9 (bitul de paritate). În cazul în care în urma verificărilor va rezulta o eroare, aceasta va fi ilustrată prin aprinderea unui led de pe placuţă. În caz contrar, codul reprezentând biţii 1-8 este transmis mai departe urmatoarei componente.

Schema bloc:

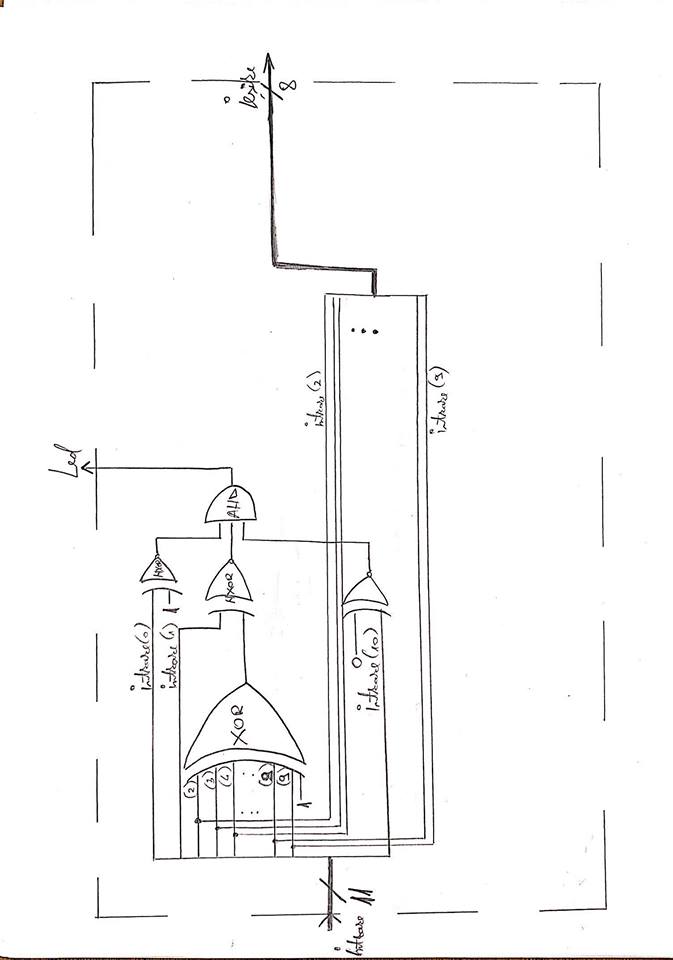


Semnificaţia notaţiilor:

•Intrare(0:10): vector pe 11 biti proveniti de la iesirea registrului de preluare

•Iesire: vector pe 8 biti reprezentand codul standard

**Schema interna:**



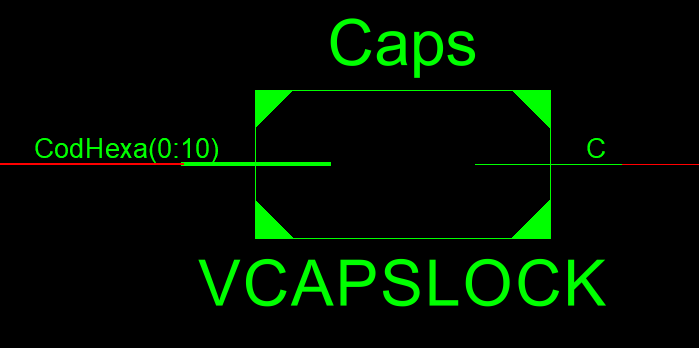
Interiorul componentei este format dintr-o poarta de tip “xor” cu 9 intrari, 3 porti de tip “nxor” pentru compararea numerelor, o poarta de tip “and” cu 3 intrari care semnifica eroare urmand ca apoi intrarile sa fie preluate de catre iesiri.

**Caps Lock:**

Proiectare componentă:

În cadrul componentei Caps Lock se va sesiza dacă de la tastatură a fost primit codul corespunzător tastei Caps Lock și se va transmite mai departe un semnal “C”, decisiv afișării.

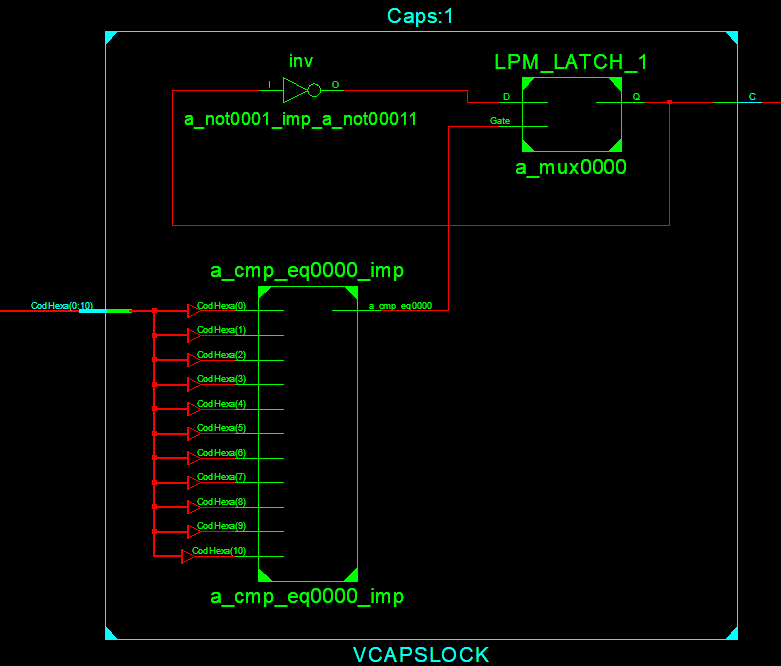
Schema bloc:



Semnificaţia notaţiior:

• CodHexa(0:10): vector pe 11 biți reprezentând codul primit de la tastatur

• C: ledul care semnalează activarea tastei de Caps Lock



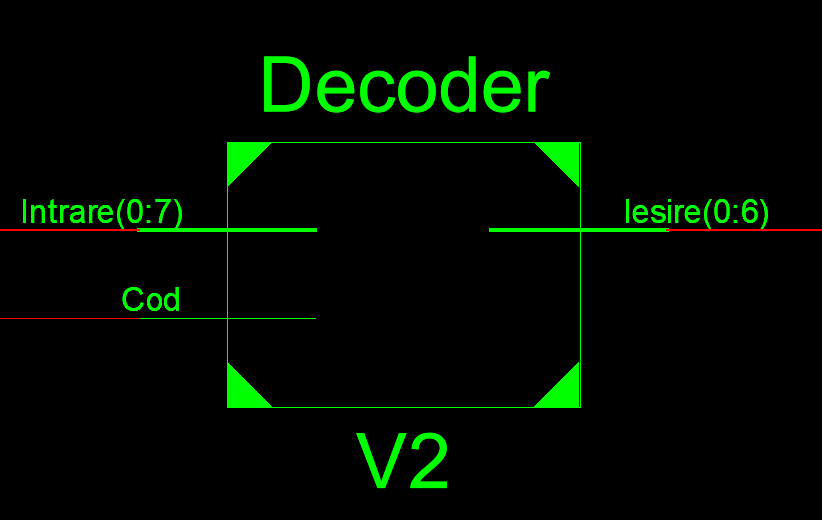
**Interiorul componentei este realizat dintr-un comparator care verifica daca a fost primit codul corespunzator tastei Caps Lock.**

**Decodificator :**

Proiectare componentă:

În cadrul decodificatorului, se va realiza corespondenţa dintre codul care vine de la tastatură şi cel de pe afişorul cu 7 segmente. Astfel ca, conform tabelului de mai sus, fiecărui cod transmis îi este atribuit un mod diferit de afişare. De asemenea, în funcţie de valoarea intrării Cod ( care are rol de Caps Lock ) decodificatorul va genera atât litere mari, cât şi litere mici. Am introdus în cod o condiţie care verifică dacă codul corespondent tastei Caps Lock este activ sau nu. În cazul în care a fost activat, atunci “Cod” va primi valoarea 1, în urma căreia decodificatorul va genera afişarea literelor mari, altfel va genera afişarea literelor mici. În cazul celorlalte taste, Caps Lock nu influenţează afişarea acestora.

Schema bloc:



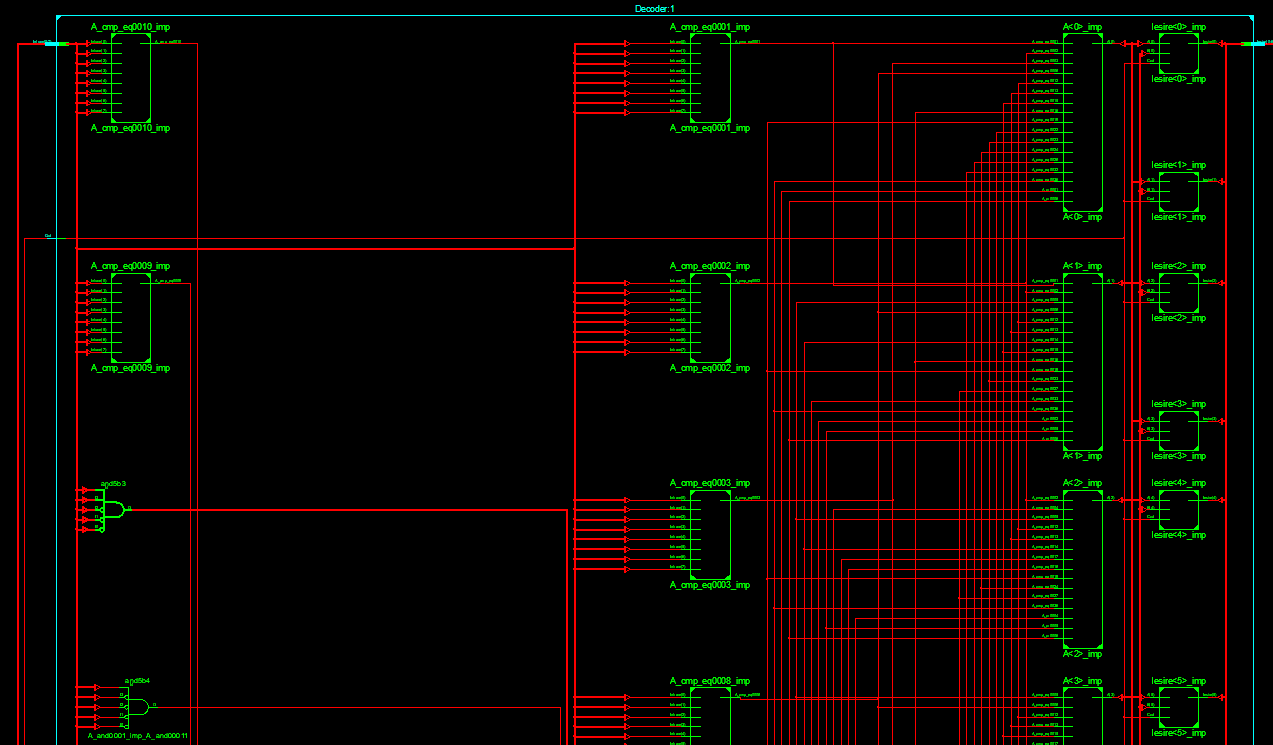
Semnificaţia notaţiior:

• Intrare(0:7): vector pe 8 biți reprezentând codul standard

• Cod: ledul care semnalează activarea tastei de Caps Lock

• Iesire: vector pe 7 biți reprezentând codul BCD pe 7 segmente

**Schema interna:**



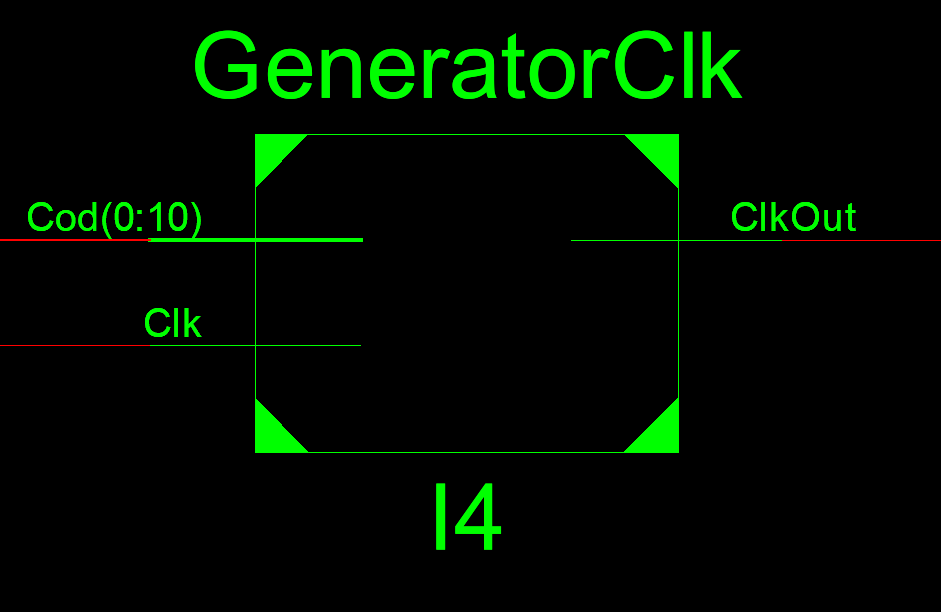
**Componenta contine in interior o serie de comparatoare, pentru a executa case-urile.**

**Generator Clk:**

Proiectare componentă:

În această secvenţă de cod, vrem să generăm clock-ul care va permite introducerea datelor în registrul de deplasare a caracterelor, respectiv în registrul de deplasare a punctelor. Orice tastă care este apăsată generează un „Make code” şi un „Break code” (de exemplu, pentru A: 1C , F0 1C) aceasta insemnând 33 de biţi (luând în considerare şi bitul de start, stop şi paritate). Însă, în cazul săgeţilor, apariţia în plus a secvenţei E0 de 2 ori, duce la 55 de biţi în total. Pentru a putea genera clock-ul în mod corect indiferent de numarul biţilor, am recurs la folosirea unui numărător care să numere de la 0 la 32, iar în momentul în care atinge această valoare, el va genera un impuls de ceas. O problemă apare doar în cazul săgeţilor, când numărul biţilor este 55. De aceea, în momentul în care se găseşte secvenţa E0, care apare la săgeţi, vom scădea 11 biţi din starea actuală a numărătorului.

Schema bloc:



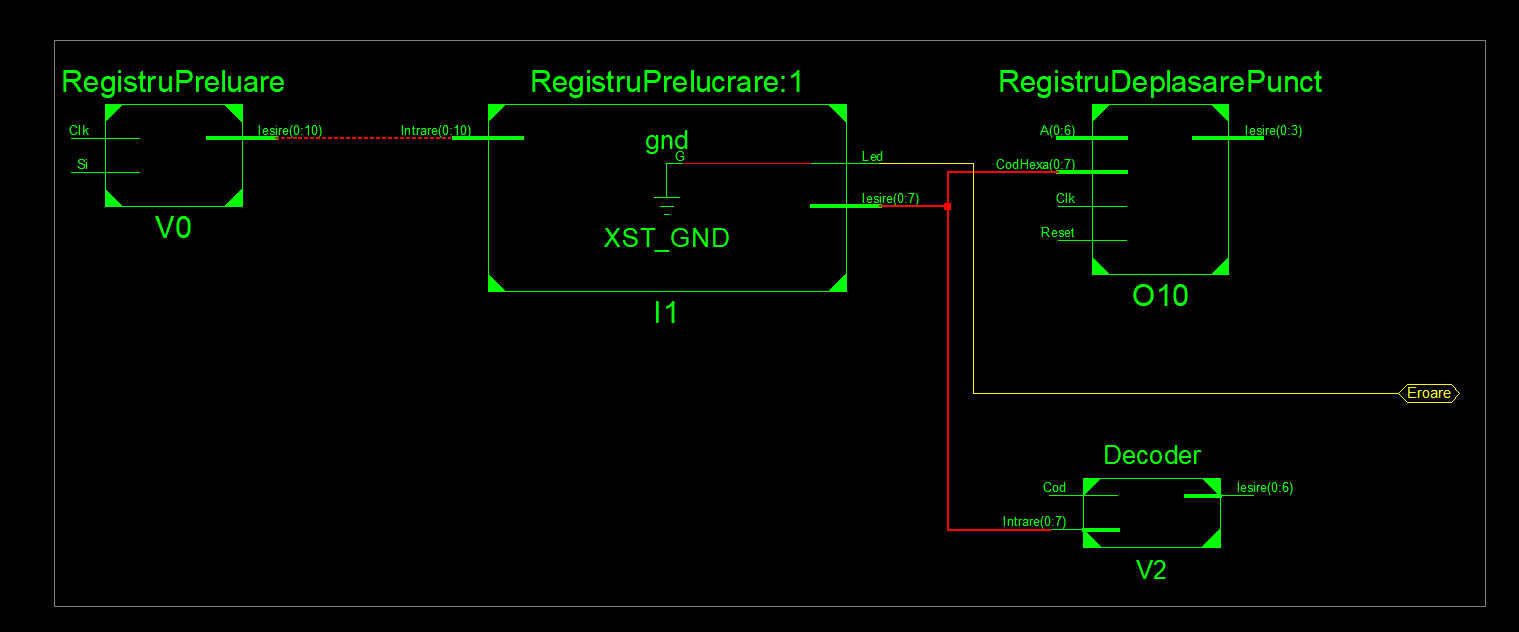
Semnificaţia notaţiilor:

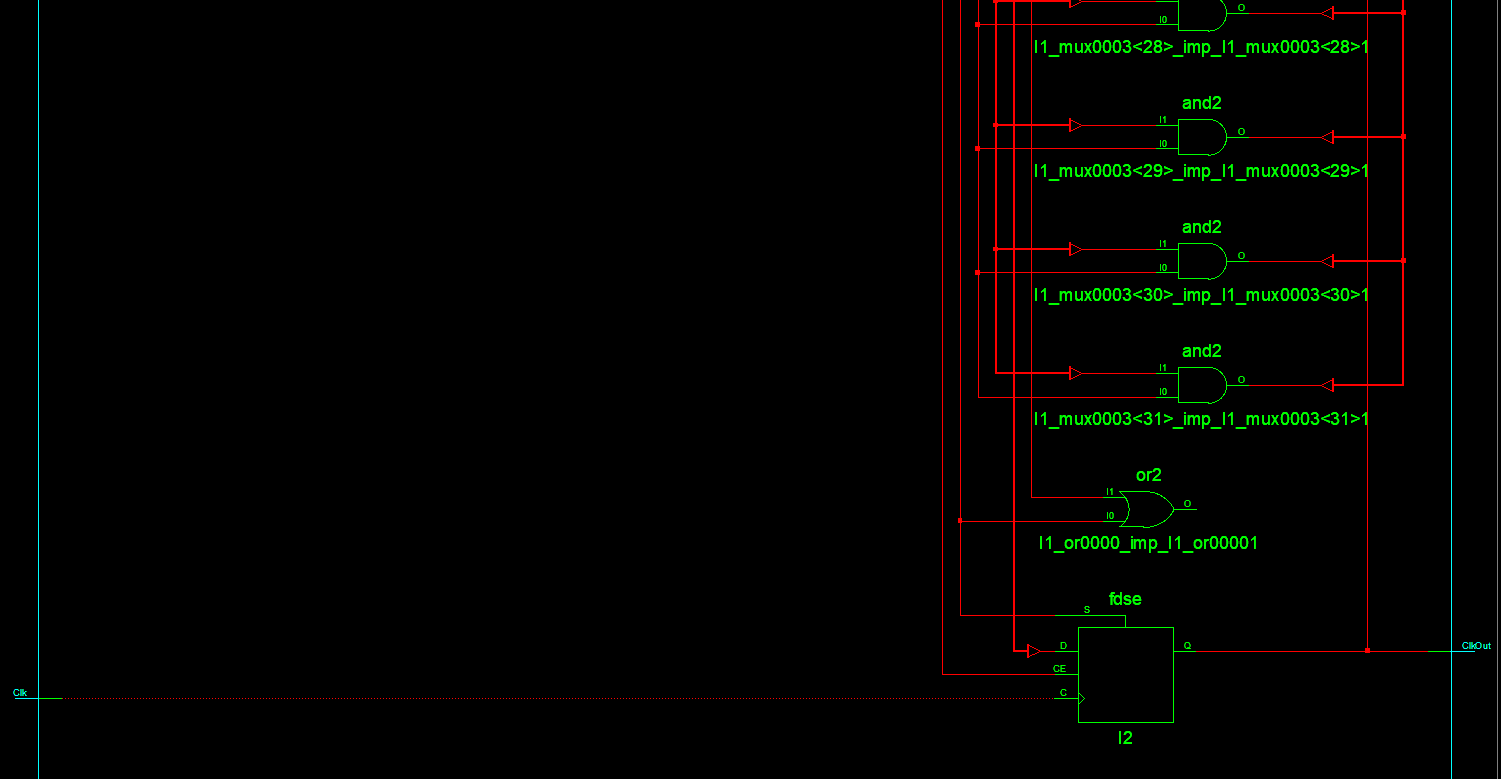
• Cod(0:10): vector 11 biți reprezentând codul preluat de registrul de preluare

• Clk: clock-ul de la tastatură

• ClkOut: clock-ul pentru un caracter nou

**Modul in care se interconecteaza componentele:**

****

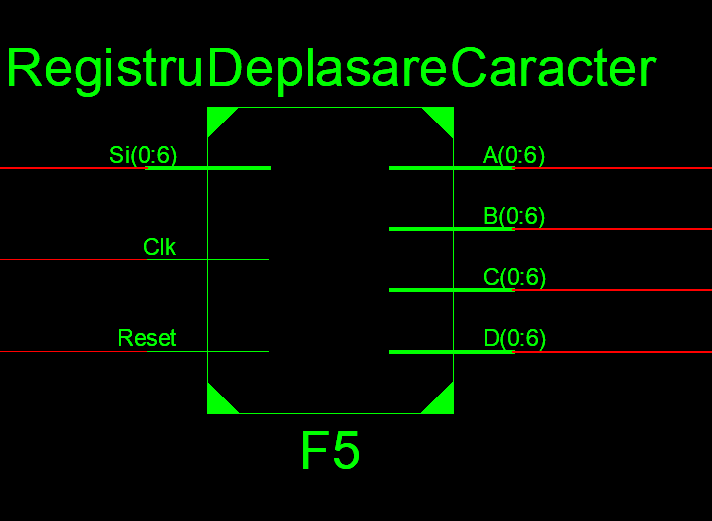
**Schema interna:**

**Registru deplasare caracter**

Proiectare componentă:

Având în vedere că se cere afişarea de caractere diferite şi se precizează că se vor vedea ultimele patru simboluri pe afişaj, este nevoie de un registru care să funcţioneze în mod FIFO, ceea ce presupune ca primul caracter citit va fi şi primul afişat. În cazul în care se introduc mai mult de 4 caractere, cel dintâi introdus se va pierde, lăsând loc noilor caractere. În momentul în care sunt apăsate tastele Caps Lock şi Enter, deplasarea caracterelor nu se mai realizează. În cazul tastei Caps Lock se realizează doar o trecere de la litere mici la litere mari, iar în cazul tastei Enter, tot afişajul se stinge, realizându-se o resetare.

Schema bloc:



Semnificaţia notaţiilor:

• Si(0:6): cod BCD pe 7 segmente

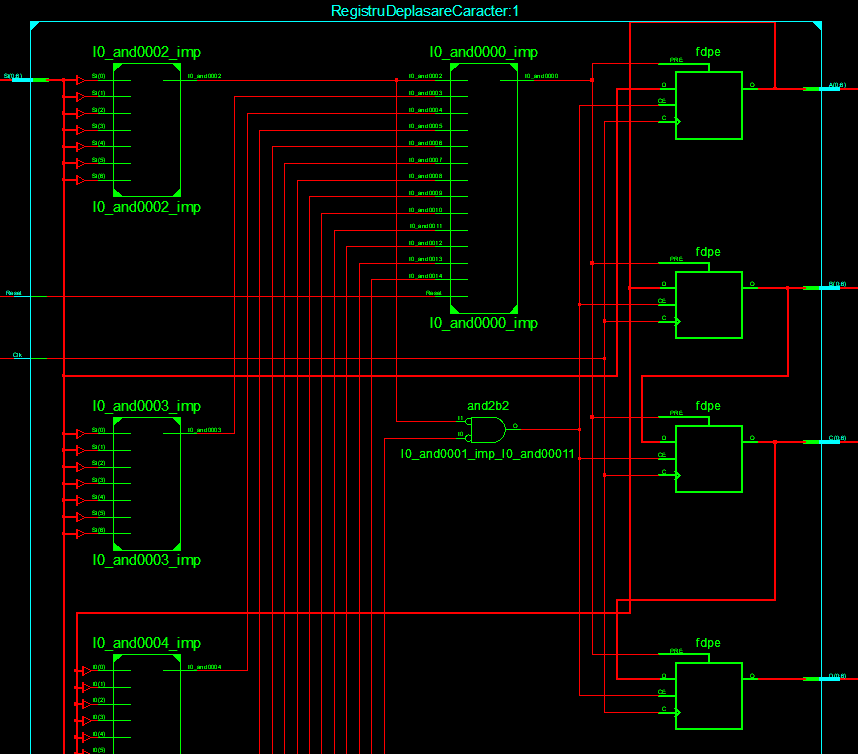
• Clk: clock-ul pentru caractere noi, provenit de la generatorul de clock

• Reset: când valoarea sa este 1, se resetează registrul

•A: ieșire ultimul caracter introdus

• A-D: ieșiri caractere în ordinea descrescătoare a introducerii lor

**Schema interna:**

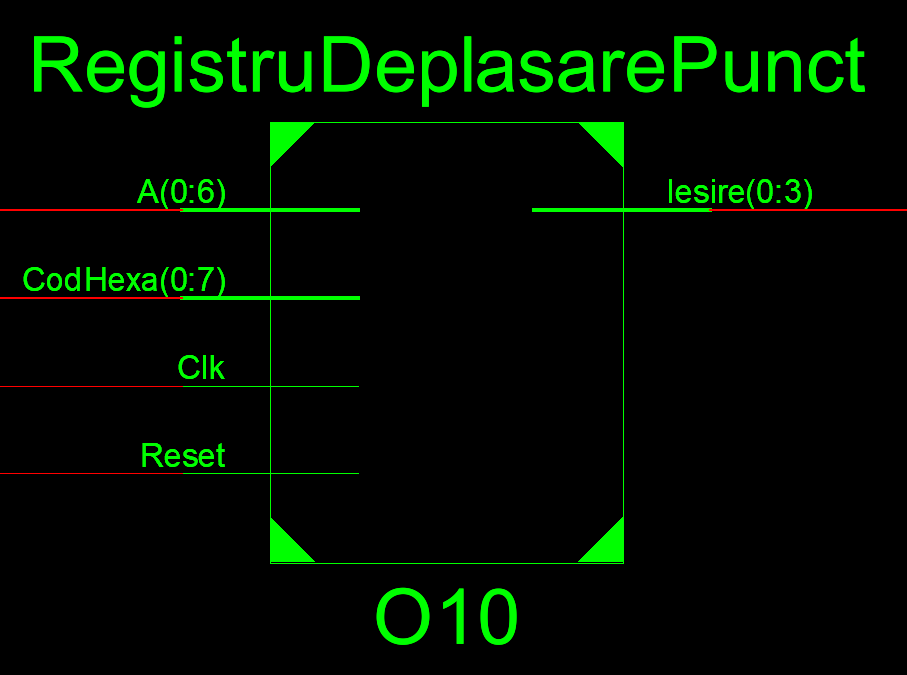


**Registru deplasare puncte**

Proiectare componentă:

Pe placuţa noastră sunt prezente şi patru puncte. Poziţia punctului activ pe 0 este controlată cu ajutorul săgeţilor stânga-dreapta. Din această cauză se va folosi din nou un registru care va controla deplasarea acestui punct, care functionează în mod asemănător cu registrul de deplasare a caracterelor. Deşi există doar 4 puncte pe afişor, noi am lucrat ca şi cum ar fi 5, deoarece prin această metodă reuşim să coordonăm mai uşor resetarea punctelor. Atunci când punctele se resetează prin Enter, ne imaginăm că cel de al 5-lea punct se activează, în timp ce celelalte puncte sunt dezactivate.

Scema bloc:



Semnificaţia notaţiilor:

• A(0:6): intrare folosită pentru corectarea unor erori si reprezintă codul BCD pe 7 segmente

• CodeHexa(0:7): pentru codul standard al săgeților

• Clk: clock-ul pentru caractere noi provenit de la generatorul de clock

•Reset: in cazul activării sale, se șterg de pe afișaj atât caracterele cât și punctul

• Iesire(0:3): ieșire registru

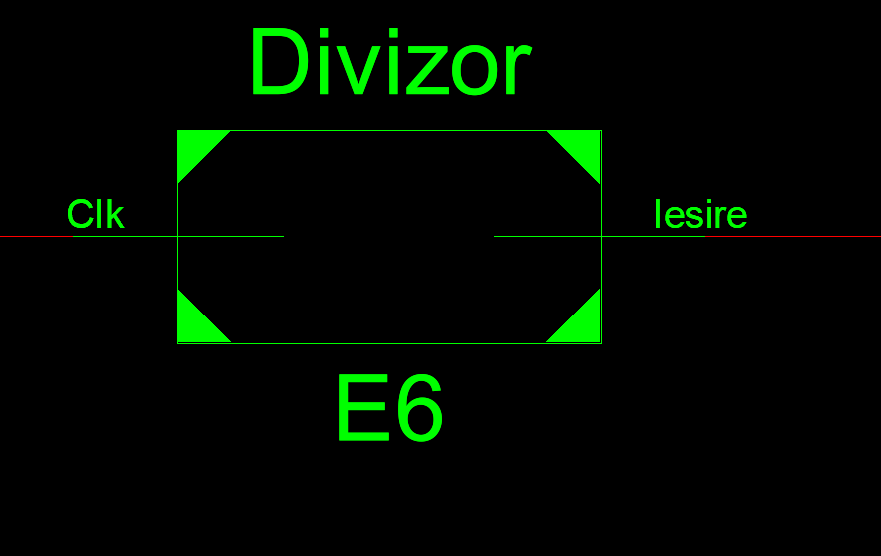
**Schema interna:**

****

**Divizor de frecvenţă**

Proiectare componentă:

Utilizatorul percepe afisarea celor 4 caractere ca pe o afisare paralela (toti cei 4 anozi sunt activi in acelasi timp). Insa aceasta presupunere este incorecta, deoarece in realitate, acestea se activeza si se dezactiveaza consecutiv la o viteza foarte mare, ceea ce creeaza impresia de continuitate. Toate acestea sunt posibile datorita divizorului de frecventa, care divizeaza clock-ul provenit de la placuta.

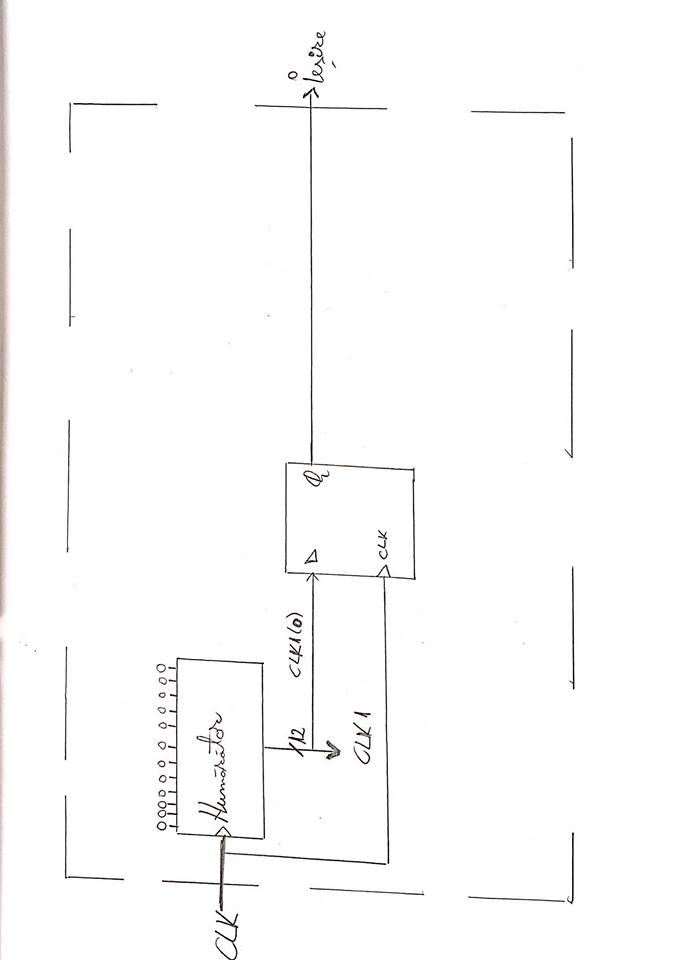
Schema bloc:

Semnificaţia notaţiilor:

• Clk: clock-ul de la plăcuță

• Iesire: clock-ul divizat

**Schema interna:**



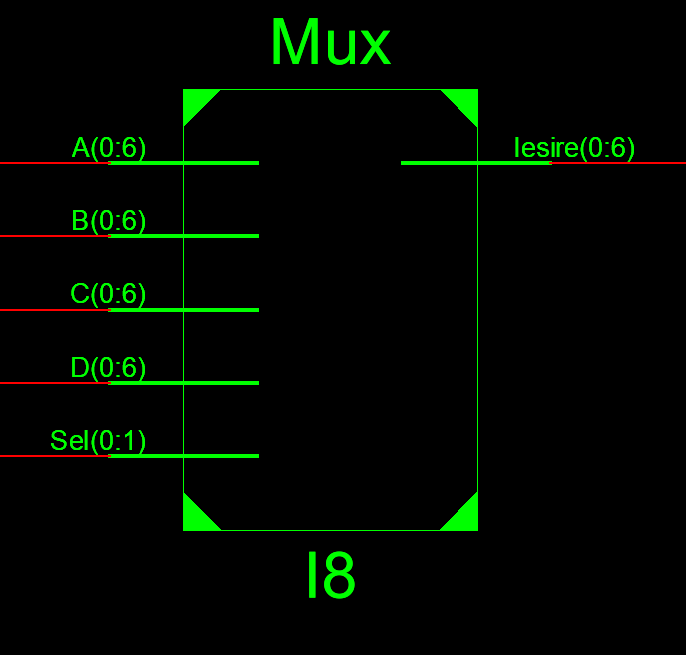
**Componenta este formata dintr-un numerator pe 12 biti, acesta fiind initializat cu 0 si un bistabil D care asigneaza iesirii bitul CLK1(0).**

**Multiplexor caractere**

Proiectare componentă:

În cadrul proiectului, am folosit două multiplexoare 4:1. Primul multiplexor are ca intrări, ieşirile de la registrul de deplasare a caracterelor, fiecare intrare fiind pe 7 biţi, aceştia reprezentând codul BCD pe 7 segmente. Cel mai nou cod introdus se va afla mereu pe cea mai semnificativă intrare a multiplexorului, astfel realizându-se încontinuu o deplasare a datelor pe intrările multiplexorului. În funcţie de selecţii, pe ieşirea multiplexorului va fi unul dintre codurile de pe cele 4 intrări, care va activa catozii corespunzători. În plus, avem nevoie de un numărător care să numere de la 0 la 3 şi ale cărui ieşiri să reprezinte selecţiile multiplexorului. Clock-ul numărătorului reprezintă clock-ul divizat.

Schema bloc:

Multiplexor

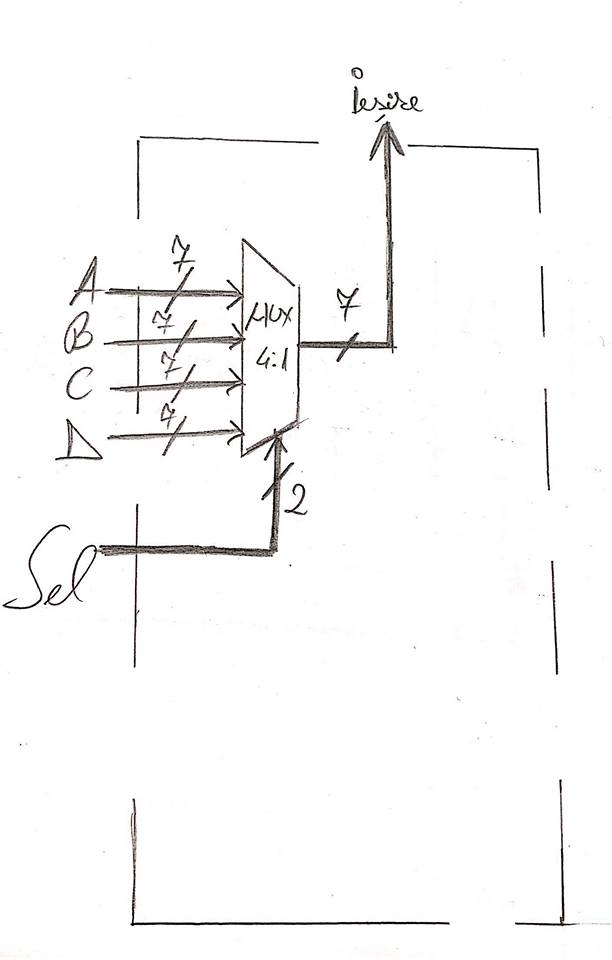
Semnificaţia notaţiilor:

A(0:6)-D(0:6): provenite de la ieșirile registrului de deplasare caractere

• Sel: selecții provenite de la numărătorul pe 2 biți

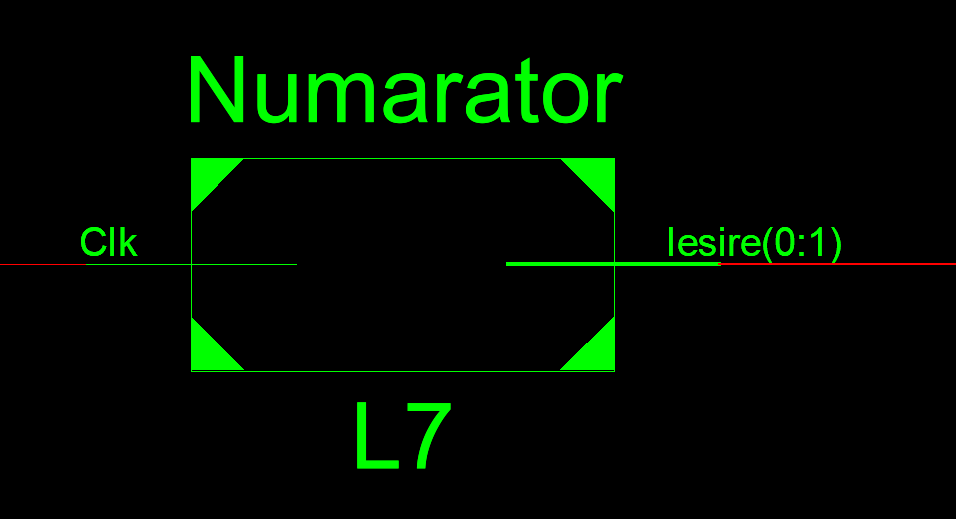
• Iesire: unul dintre caractere de pe intrări, în funcție de selecții

**Schema interna:**

****

**Interiorul componentei este simplu realizat print-un multiplexor simplu 4:1 cu cale de intrare pe 7 biti, selectie pe 2 biti si iesirea deasemenea pe 7 biti, acesta fiind capabil sa aleaga caracterul care se doreste a fii afisat.**

**Numarator**

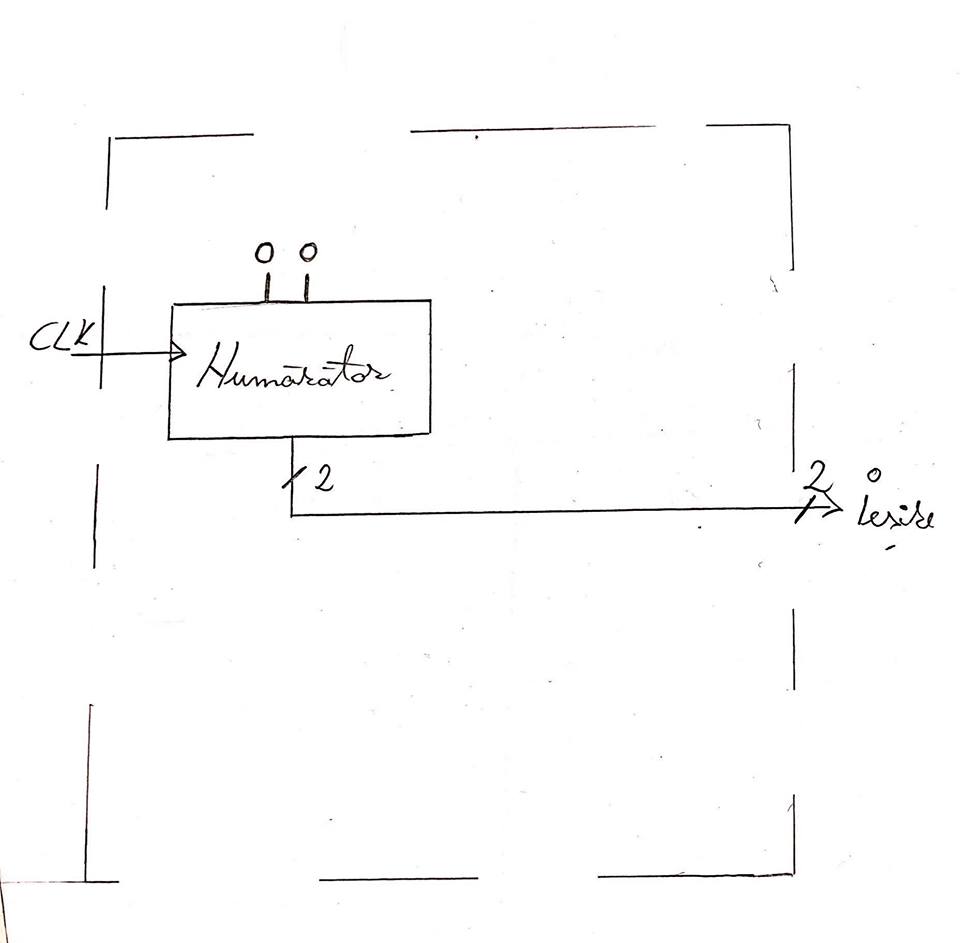
****

Semnificaţia notaţiilor:

• Clk: reprezintă clock-ul divizat, provenit de la divizorul de frecvență

• Iesire: ieșiri care reprezintă selectiile pentru multiplexoarele folosite

**Schema interna:**

****

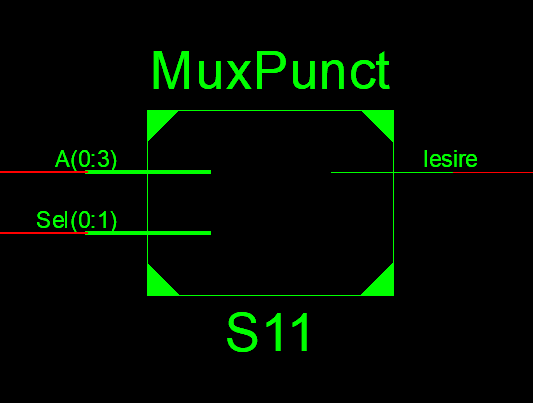
**Si aceasta componenta esti simplu realizata, printr-un numarator.**

**Multiplexor punct**

Proiectare componentă:

Pentru deplasarea punctului avem nevoie de un multiplexor, care se va baza pe selecţiile aceluiaşi numărător, amintit în descrierea multiplexorului de la afişarea caracterelor. Astfel, diferenţa dintre cele două multiplexoare constă în faptul că aici, intrările nu sunt pe 7 biţi,ci doar pe câte un bit.

Schema bloc:



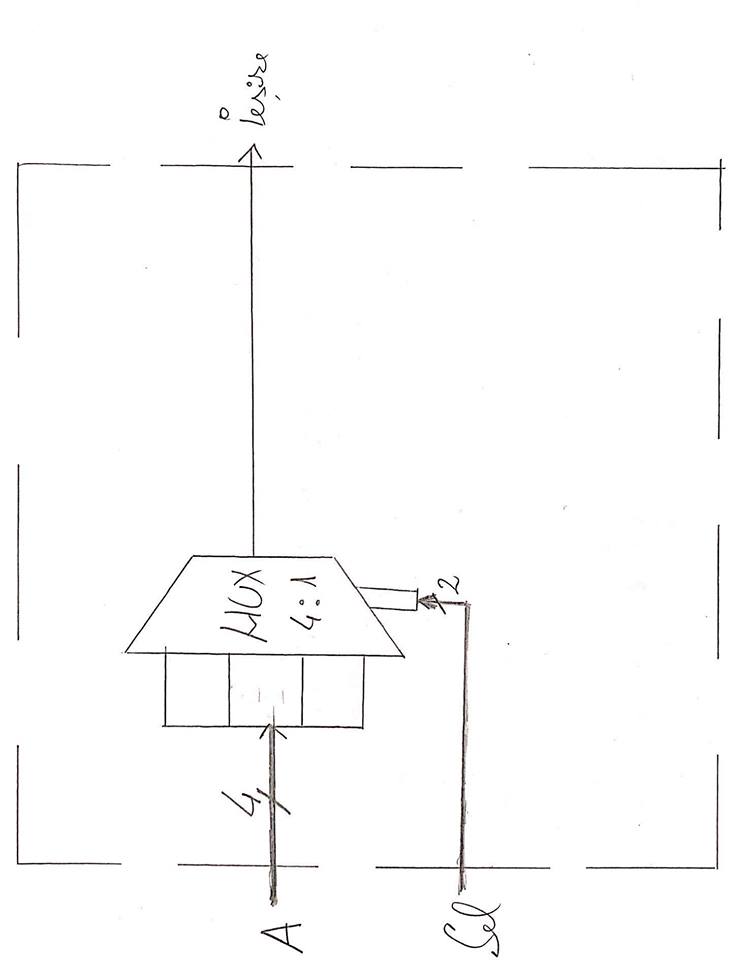
Semnificaţia notaţiilor

• A(0:3): provenită de la registrul de deplasare punct

• Sel(0:1): selecții, provenite de la un numărător pe 2 biți, asemenea multiplexorului pentru caractere

• Iesire: ieșire pe 1 bit, reprezintă aprinderea unuia dintre puncte

**Componenta contine in interior un multiplexor 4:1, cu calea de date pe 1 bit, alegand mai departe bitul care doreste a f ii asignat iesirii.**

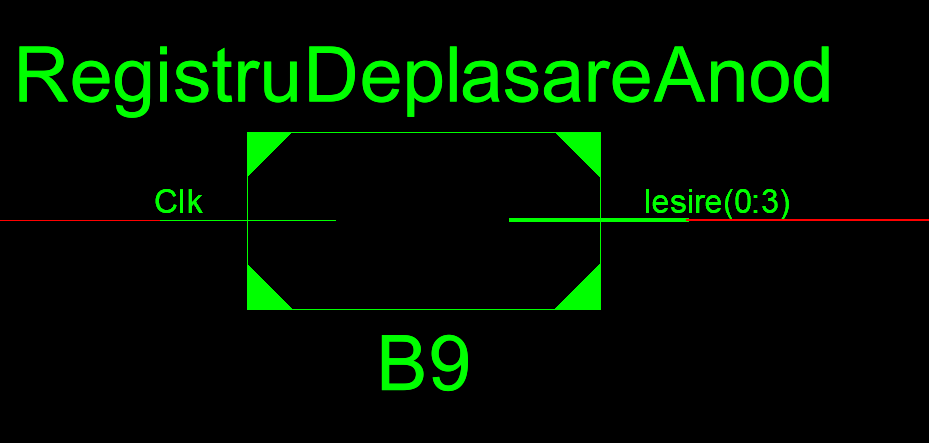
**Schema interna:**

**Registru deplasare anozi**

Proiectare componentă:

Pentru afişarea unor caractere diferite pe cele 4 afişoare, folosim un registru pe 4 biţi care va funcţiona astfel: “1110”->”1101”->”1011”->”0111”->”1110” etc, unde cei 4 biţi reprezintă cei 4 anozi, care sunt activaţi pe rând, în mod ciclic, toate acestea petrecându-se la viteză mare. Astfel, este posibilă afişarea caracterelor diferite pe anozi.

Schema bloc:

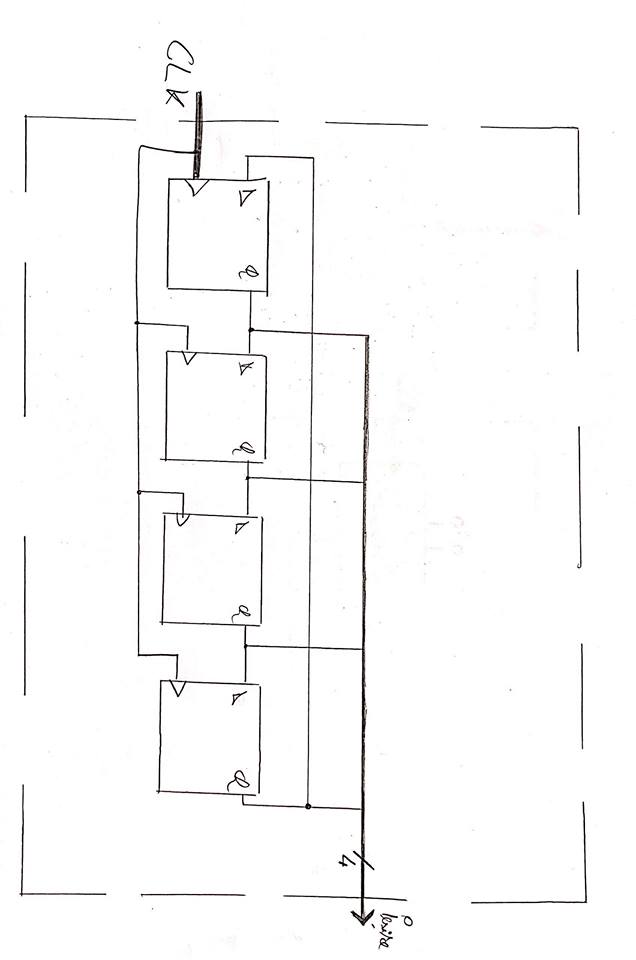


Semnificaţia notaţiilor:

Clk: Clock divizat provenit de la divizorul de frecvență.

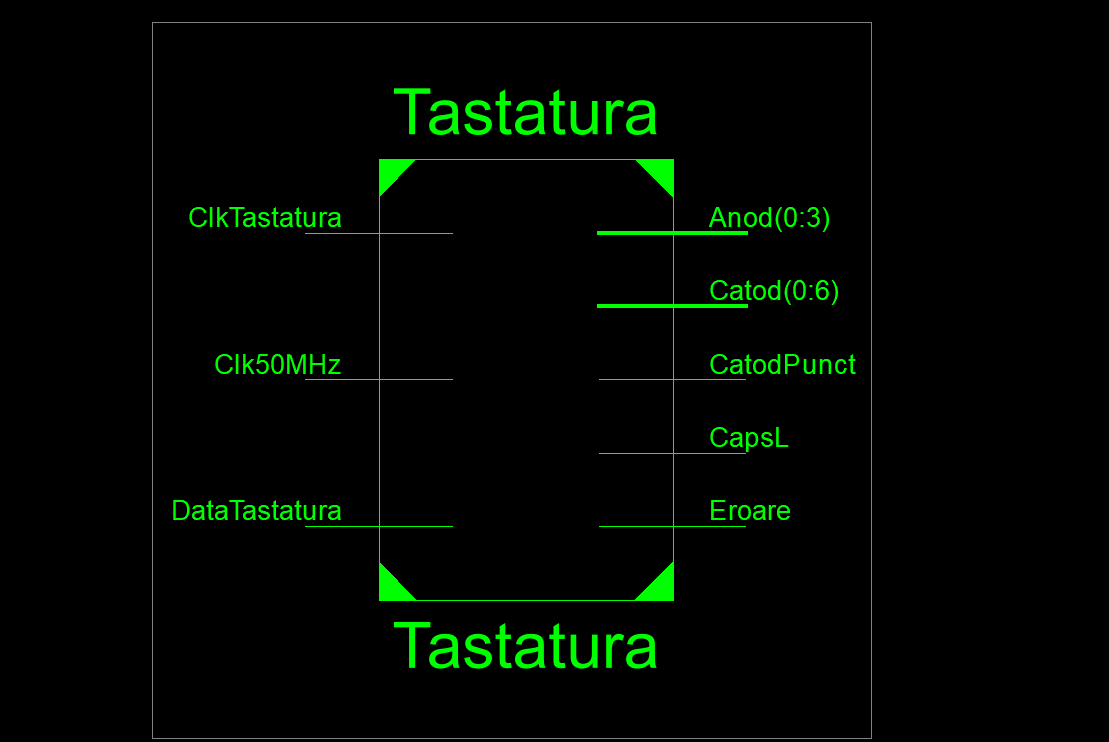
Ieșirea: Iesire(0:3): vector reprezentând cei 4 anozi.

Schema interna:



**Componenta are in interior un registru de shiftare ciclic, format din 4 bistabile de tip D fiind initializat la “1110”.**

**Proiectarea tastaturii:**



Clk50MHz: clock-ul provenit de la plăcuță

DataTastatura: biții standard ai fiecărei taste

ClkTastatura: clock-ul provenit de la tastatura

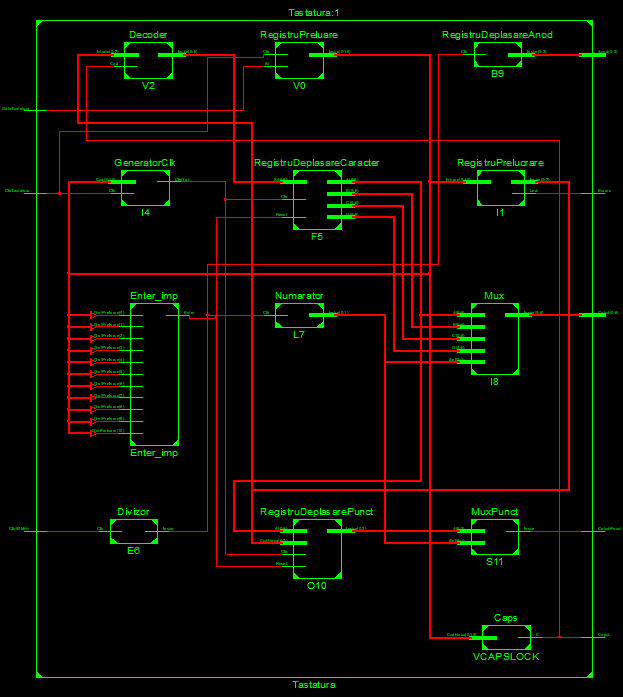
Anod(0:3): afișarea pe anod

Catod/Catodpunct: afișare pe catod/punct

CapsL: reprezintă caps lock-ul

Eroare: se folosește în cazul unei erori

Schema interna:



**Pe langa componentele vizibile, schema interioara necesita si un "Enter\_imp",care verifica daca a fost primit codul pentru resetare(Enter).**

**Justificarea soluţiei alese**

La realizarea acestui proiect am optat pentru o modalitate ușor de înțeles de către oricine, de aceea am folosit denumiri sugestive în cadrul întregului proiect și în același timp în cadrul acestei docmentații am înșirat și semnificația fiecărei notații în parte.

În legatură cu organizarea programului s-a folosit o abordare modulară, deoarece astfel programul este mai ușor de înțeles și de urmărit. Împărțirea pe module este una practică, ușor de înțeles si părțile componente au nume sugestive. Nu este nevoie de păstrarea comenzilor efectuate în memorie, ele fiind procesate instant. Placa este setatǎ sǎ preia date în mod continuu fǎrǎ a fi necesarǎ o resetare dupǎ fiecare transmisie, iar ca modalitate de afișare s-au folosit 4 afișoare 7-segmente, pentru afișarea corespunzătoare a literelor primite de la tastaturǎ.

Metoda utilizată de noi impune încă de la început anumite condiții pe baza cărora utilizatorul să fie sigur că codul preluat este corect și potrivit pentru afișare. Condiția constă în simpla verificare a corectitudinii bitului e paritate, bitului de start și de stop. În cazul în care nu se sesizează nici o eroare se transmit mai departe cei 8 biți rămași.

Pe parcursul implementării am sesizat că apăsarea anumitor taste generează resetarea întregului afișor. Din această cauză am fost nevoiti să introducem niște condiții, care să nu permită resetarea, doar în cazul în care se apasă tasta ENTER. Aceaste condiții au fost introduse în cadrul registrului de deplasare caractere și de deplasare al punctului.

**Utilizare şi rezultate**

*Pași necesari pentru utilizare și rezultate*

Pentru crearea, compilarea și simularea codului VHDL se utilizează limbajul de descriere hardware Active-HDL, iar pentru a putea exporta proiectul pe placa FPGA se utilizează mediul de dezvoltare Xilinx ISE.

În cadrul proiectului primul pas făcut de către utilizator este de a cupla toate componentele fizice între ele. Știind că codul din proiectul prezent este corect, utilizatorului nu-i rămâne sarcina de a scrie cod, doar de a-l folosi. După exportarea proiectului pe placa FPGA și utilizarea mediului de dezvoltare XILINX ISE utilizatorul poate folosi tasta. Utilizatorul introduce datele folosind tastele, acestea urmând să fie afișate pe cele 7 segmente ale plăcuței. În momentul apăsării unei taste noi are loc o shiftare spre stânga a tuturor caracterelor afişate până în acel moment, ultima valoare fiind pierdută. În cadrul proiectului se ține cont de tasta ENTER, la apăsrea ei se resetează afișoarele. O altă tastă specială este CAPS LOCK-ul, apăsarea căreia utilizatorului îi va fi permis să scrie cuvintele cu majuscule.

Resurse utilizate:

Manual VHDL Advanced Digital Design with Basys 2

<http://www.computer-engineering.org/ps2keyboard/>

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

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

<http://www.electronics-tutorials.ws/combination/comb_6.html>

<https://en.wikipedia.org/wiki/Seven-segment_display>

**Posibilități de dezvoltare ulterioară**

Controllerul de tastatură PS2 poate să fie îmbunătăţit prin adăugarea de noi funcţionalităţi:

Includerea tastei special CTRL. Această tastă se poate folosi în combinație cu tastele C și V. În urma acestor combinații se va efectua copierea secvenței de pe afișor, respectiv inserarea secvenței copiate. Introducând o secvență pe afișor apoi apăsând combinația CTRL+C, această secvență se va memora ca în viitor să se poată reintroduce pe afișor doar prin apăsarea combinației CTRL+V, indifferent de numărul de caractere introduse pe afișor intre aceste două operații.

Introducerea printre tastele folosite a tastei TAB. Prin apăsrea acestei taste se va inversa direcția de shiftare a datelor. Dacă până acum datele s-au shiftat spre stânga, după apăsarea tastei TAB shiftarea se va realiza spre dreapta.