Proiect la Structura Sistemelor de Calcul

Proiect:

Comunicația dintre placa Nexys 4 DDR și o tastatură USB HID

Nume profesor indrumator: Nume Student: Vasilache Maria

Lisman Florin Grupa: 30238

-2022-

Contents

[1. Rezumat 4](#_Toc118033558)

[2. Introducere 4](#_Toc118033559)

[3. Fundamentare teoretica 5](#_Toc118033560)

[*1.2.2 USB HID Host* 5](#_Toc118033561)

[1,2.3 HID Controller 5](#_Toc118033562)

[*1.2.4 Bus Timing* 5](#_Toc118033563)

[4. Proiectare şi implementare 6](#_Toc118033564)

[*1.2.5 Conectarea tastaturii la FPGA* 7](#_Toc118033565)

[*1.2.6 Tastatura-codificare taste* 8](#_Toc118033566)

[1.2.7. Codificare taste 8](#_Toc118033567)

[*1,2.8. PS2-KeyBoard* 9](#_Toc118033568)

[*1.2.9. Afisare SSD( seven segment display, afisor sapte segmente)* 10](#_Toc118033569)

[1.3 Schema bloc 10](#_Toc118033570)

[1.3. Codificare cod ASCII pentru toate tastele: 13](#_Toc118033571)

[*1.3.1 Taste speciale* 13](#_Toc118033572)

[*1.3.2 Paranteze:* 13](#_Toc118033573)

[*1.3.3 Litere MARI – mici* 13](#_Toc118033574)

[*1.3.4 Cifre-codificare* 14](#_Toc118033575)

[*1.3.5. Afisare pe SSD- Reprezentare grafica: Codificare:hgfedcba* 14](#_Toc118033576)

[*1.3.6 Afisare pe SSD-Schema bloc* 16](#_Toc118033577)

[*1.3.7. Codificare VHDL* 17](#_Toc118033578)

[*1.3.8 Afisare pe SSD- Codificare VHDL* 17](#_Toc118033579)

[2. Implementare in VHDL 18](#_Toc118033580)

[2.1 Implementare debouncer 18](#_Toc118033581)

[*2.2* Implementare Afisare taste apasate 19](#_Toc118033582)

[2.3 Controller: PS/2 19](#_Toc118033583)

[2.3.4. Generare intrerupere 21](#_Toc118033584)

[2.4.1. Controlarea tastelor apasate=>generarea codurilor ASCII 21](#_Toc118033585)

[2.4.2. Retinerea tastelor apasate 21](#_Toc118033586)

[4.Legarura intre cod si componente: 21](#_Toc118033587)

[5. Rezultate experimentale 22](#_Toc118033588)

[6. Concluzii 25](#_Toc118033589)

[Bibliografia: 26](#_Toc118033590)

# 1. Rezumat

Proiectarea proiectului se realizeaza utilizând limbajul VHDL. Intrucat tema impune conectarea unei tastaturi la placuta urmand sa fie apasata la un moment dat o tasta, care, se va afisata tasta apăsate pe afișajul cu 7 segmente al plăcii Nexys 4 DDR.

Tema se va implementare pe placa Digilent Nexys 4 DDR.

Pentru acest proiect este necesara cunoaserea modului in care se conecteaza tastatura la Nexys 4. De asemenea, etapa de identificare a tastei apasate, urmand sa se afiseze valoarea acesteia pe SSD.

Tastatura folosește protocolul standard IBM pentru a comunica cu computerul. Acest protocol are scopul de a trimite *scan code-ul* tastei care este apăsată către PC și de a primi o comandă de răspuns de la acesta. Astfel, avem de-a face cu un protocol bi-direcţional deoarece fiecare componentă(PC/tastatură) trimite și primește comenzi

Pasii pe care trebuie sa-i urmam sunt:

1. Conectarea placutei la tastatura prin interfața PS2
2. Codificarea tastei apasate
3. Decodificarea tastei
4. Afisarea pe SSD a valorii corespunzatoare

# 2. Introducere

Proiectul presupune transmisia de date preluate de la tastatura, decodificare si afisarea pe SSD. Intrucat avem doar 7 segmente, este imposibila afisarea tuturor literelor existente, insa pentru o implementare cat mai interactiva, voi afisa pe SSD atat litere mari, cat si litere mici.

Modul de functionare, respecta modelul real, iin care pentru afisarea literelor mari se apasa simultan tastele Shift + Tasta.

De altfel, pentru utilizatorii care folosesc in mod frecvent tasta CapsLock, pentru litere mari, am pus la dispozitie existenta acste functionalitati, prin urmare, se pot folosi de ele.

De remarcat este faptul ca nu s-au putut afisa toate literele mari, din acest motiv, se va afisa 0 pe SSD daca nu exista reprezentare.

Limitarea este data de numarul de afisaje digitale. Pentru o implementare viitoare, care ar pune la dispozitie un SSD mai generos, codul poate identifica apasrea oricarei litere sau combinatii, iar in viitor se pot modifica liniiile de cod ale reprezentariii pe Anozi.

# 3. Fundamentare teoretica

Pentru analizarea cerintei, a fost nevoie de o documentare atenta, suportul pe care l-am luat ca punct de referinta, a fost documentatia de la Digilent.

In aceasta sursa am gasit informatii cu privier la PS/2, iar prin prisma aprofundarii acestui subiect, mi-am facut o imagine de ansamblu cu privier la pasii pe care trebuie sa-I fac pentru a contura proiectul

Protocolul va prelua comunicarea . Acest lucru se face folosind liniile de date și de clock . Aceste linii sunt **HIGH** atunci când nu are loc comunicarea(idle). Comunicarea tastatură-gazdă și gazdă-tastatură sunt ușor diferite, astfel vor fi explicate separat . Rețineți că, frecvența ceasului este de aproximativ 20-30Khz și este generată de tastatură , dar numai atunci când o transmisie are loc .

Tastatura este liberă să trimită date către gazdă , atunci când ambele linii de date și de clock sunt ținute **HIGH** . Tastatura va lua nivelul **LOW** de date ( Bitul de start ) și apoi începe să genereze impulsuri de ceas de pe linia de clock

1.2. Descrierea componentelor utilizate(din cadrup placutei)

# 1.2.2 USB HID Host

Placa este prevazuta cu USB HID Host, odată ce FPGA este programat, microcontrolerul trece în modul aplicație, care este USB HID Host în acest caz. Firmware-ul din microcontroler poate conduce mouse-ul sau o tastatură atașată la conectorul USB.

Sunt acceptate doar tastaturile și mouse-urile care acceptă interfața BootHID.

# 1,2.3 HID Controller

Microcontrolerul cu funcție auxiliară ascunde protocolul USB HID de FPGA și emulează un PS/2 bus de stil vechi. Microcontrolerul se comportă exact ca o tastatură sau un mouse PS/2. Aceasta înseamnă că noile modele pot reutiliza nucleele PS/2 IP existente. Mouse-urile și tastaturile care utilizează protocolul PS/2 folosesc o magistrală serială cu două fire (ceas și date) pentru a comunica cu o gazdă.

Pe Nexys4 DDR, microcontrolerul emulează un dispozitiv PS/2, în timp **ce FPGA joacă rolul gazdei**. Atât mouse-ul, cât și tastatura folosesc cuvinte **de 11 biți** care includ un bit de pornire, un octet de date (LSBfirst), paritate impară și un bit de oprire, dar pachetele de date sunt organizate diferit,

Interfața tastaturii permite **transferuri de date bidirecționale** (deci dispozitivul gazdă poate aprinde LED-urile de stare de pe tastatură).

# 1.2.4 Bus Timing

Diagram, schematic

Description automatically generated

Ceasul și semnalele de date sunt conduse numai **atunci când au loc transferuri de date**; în caz contrar, ele sunt menținute în starea de inalta impedanta (driver-driver deschis).

Acest lucru necesită ca atunci când semnalele PS/2 sunt utilizate într-un design, pull-up-urile interne trebuie să fie activate în FPGA pe pinii de date și de ceas. (Pull-up: **1, cand nu** e apasat, **‘0’ cand e** apasat.

Semnalul ceasului este în mod normal condus de dispozitiv, dar poate fi ținut la nivel scăzut de gazdă în cazuri speciale. Timer-ele definesc cerințele de semnal pentru comunicațiile mouse-gazdă și **comunicațiile bidirecționale de la tastatură**. Un circuit de interfață PS/2 poate fi implementat în FPGA pentru a crea o interfață de tastatură sau mouse.

# Proiectare şi implementare

# 5.1. Descrierea prin diagrama de stare

Se poate vedea evolutia autommatului, pana la start pana la ultima, fiind precizate cele mia importante semnale de comanda.



FSM – pentru tastatura

Diagram

Description automatically generated

# 1.2.5 Conectarea tastaturii la FPGA

Când o tastatură sau un mouse **este conectat** la Nexys4 DDR, se trece la **“self-test passed",** comanda "**0xAA**" este setată la gazdă.

După aceasta, pot fi emise comenzi către dispozitiv. Deoarece atât tastatura, cât și mouse-ul folosesc același port PS/2, se poate spune tipul de dispozitiv conectat folosind ID-ul dispozitivului.

# 

# 1.2.6 Tastatura-codificare taste

PS/2 foloseste coduri atunci cand se apasa o tasta, codificarea se face in cod ASCII, care este unic pentru fiecare tasta.

Daca tasta este **held-down**, codul va fi transmis in mod repetat la fiecare **100ms.**

Cand se apasa o tasta, se transmite codul F0 key-up code, urmat de codul tastei propriu-zise.

Cand shift este apasata, se produce un nou caracter(litera mare), Majuscula trimite un cod additional

Pentru ca gazda(placuta) **este conectata la ceas**, se verifica in permanenta daca s-a apasat o tasta, mai precis, daca s-a detecta**t un cod specific tastelor**

# 1.2.7. Codificare taste

**Frecventa**: 20-30KHz

Tastatura trimite **date** pe **11 biti:**

**🡪bit de start**: ‘0’

**🡪8 biti:** codul scanat

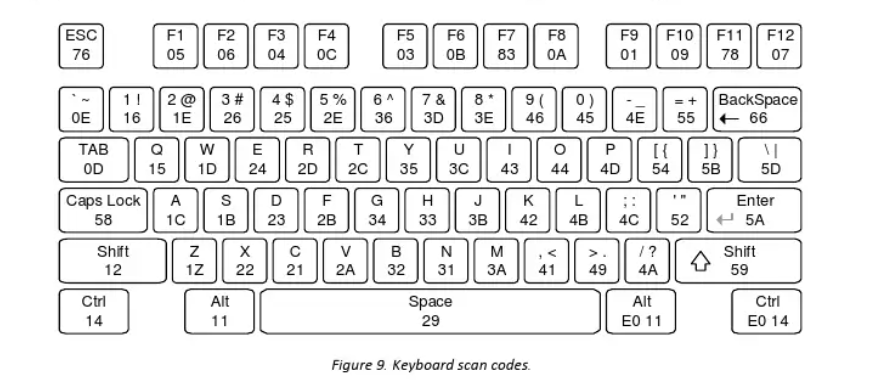
**🡪1 bit:** de paritate (impara)

**🡪1 bit:** bit de stop

Tastatura genereaza 11 tranzitii(20-30 KHz).

Datele sunt validate pe: **falling edge** al ceasului. In cod VHDL, vom folosi urmatoarea sintaxa:

**falling\_edge(clk)**



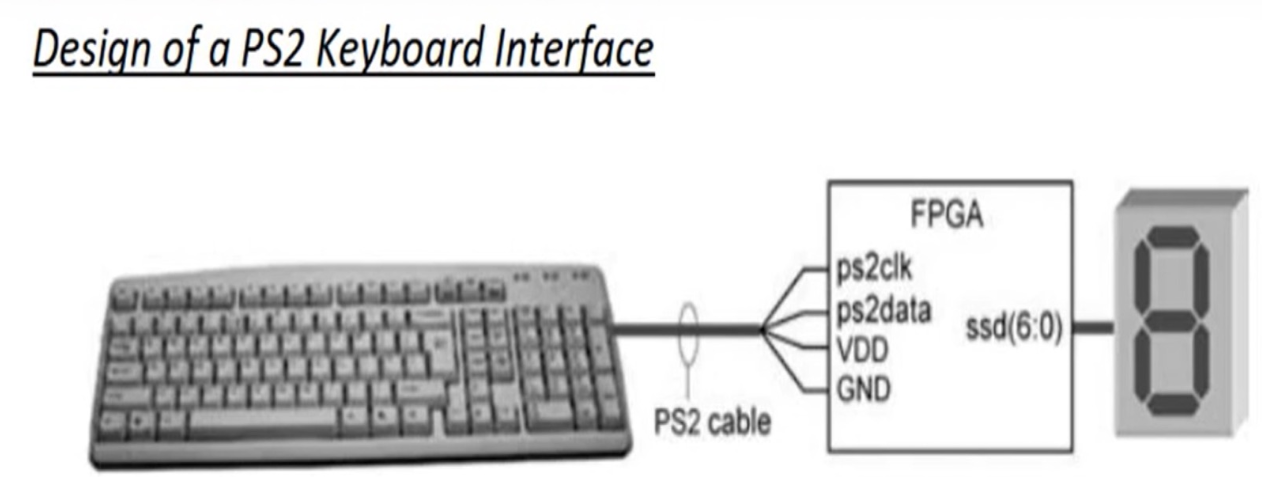
# 1,2.8. PS2-KeyBoard

Aceasta este o componenta importanta pentru ințelegeți comunicarii computerele cu perifericele, mai cu seama comunicarea calculator-tastatura.

Pentru interfața PS2, codul se va implementa folosind limbajul VHDL.

Din punct de vedere al aparitiei Interfața PS2, aceasta a fost introdusă de IBM în 1987 pentru conectarea computerelor la dispozitive cu tastatură și mouse. Chiar dacă acum dă loc interfeței USB (universal serial bus), studiul său ajută la înțelegerea modului în care computerele comunică cu perifericele.

Sistemul este reprezentat în figură, care arată o tastatură PS2 conectată la un FPGA, care, la rândul său, alimentează un SSD.



Dorim să proiectăm un circuit capabil să citească tastele numerice (0, 1, ... , 9) si litere de la tastatură și să le afișăm pe SSD atunci când aceste taste sunt apăsate.

1.2.8 Schema bloc PS/2 + intrari explicate

Diagram

Description automatically generated

Text

Description automatically generated

# 1.2.9. Afisare SSD( seven segment display, afisor sapte segmente)

Placa dispune de 8 afisaje, insa acestea, au catod comun, iar anodul este separat, prin urmare, avem 4 anozi si 8 catozi. Atat catodul, cat si catodul, sunt active pe ‘0’. Din acest motiv, pentru codificare, se va folosi logica negative pentru activarea unui semnal.

Aceste afisaje nu se pot aprinde simultan, avand anodul separate, de aceea se programeaza la o frecveenta de 62,5 Hz, cu o rata de refresh la 16ns, fiecare cifra fiind active 2ns. Acest mecanism va fi perceput de ochiul uman ca o afisare simultana a cifrelor.

Diagram

Description automatically generated

# 1.3 Schema bloc

Observatie: FPGA-ul este bidirectional, acesta va primi comenzile de la apasarea unei taste, urmand sa afiseze pe SSD valoarea acesteia

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

Observatie:

Cand tastatura este conectata, se va transmite de la reciver, codul 0xFF, care semnifica faptul ca este in staraea de primire a semnalului. Cand Transmiter-ul va detecta codul 0xFF, se va transmite codul de activare “0xAA”, care semnifica faptul ca se pot receptiona coduri de la taste, respective combinatii de taste.

Dupa aceasta initializare, se vor putea accepta coduri pentru CapsLock si pentru Ctrl, Alt…

# 1.3. Codificare cod ASCII pentru toate tastele:

# Observatie, intrucat afisajul dispune de 8 segmente, nu se vor putea reprezenta toate tastele, de aceea am ales doar tastele ce urmeaza sa fie afaisate, insa pentru documentare, am pus codurile pentru toate tastele.

# 1.3.1 Taste speciale

|  |
| --- |
| x"12" - left shift code  x"59" -right shift code |

# 1.3.2 Paranteze:

x"54"--^[

x"5B" --^]

x"4E"--\_

x"4E" --^\_

# 1.3.3 Litere MARI – mici

x"1C" --A/a

x"32"--B/b

x"21"--C/c

x"23"--^D/d

x"24" -^E/e

x"2B" --^F/f

x"34"--^G /g

x"33" --^H/h

x"43"--^I/i

x"3B" --^J/j

"42" --^K/k

x"4B" --^L/l

x"3A"--^M/m

x"31" --^N/n

x"44"--^O/o

x"4D"--^P/p

x"15" --^Q/q

x"2D"--^R/r

x"2C"--^T/t

x"3C"--^U/u

x"2A"--^V/v

x"1D" --^W/w

x"22"--^X /x

x"35"--^Y/y

x"1A"--^Z/z

# 1.3.4 Cifre-codificare

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 16 (F016) |  |  |
| 2 | 1E (F01E) |  |  |
| 3 | 26 (F026) |  |  |
| 4 | 25 (F025) |  |  |
| 5 | 2E (F02E) |  |  |
| 6 | 36 (F036) |  |  |
| 7 | 3D (F03D) |  |  |
| 8 | 3E (F03E) |  |  |
| 9 | 46 (F046) |  |  |
| 0 | 45 (F045) |  |  |

# 1.3.5. Afisare pe SSD- Reprezentare grafica: Codificare:hgfedcba

Diagram

Description automatically generatedA: 0000010(1)

**Diagram

Description automatically generated**B: 1100000(1)

Diagram

Description automatically generatedC:0110001(1)

Diagram

Description automatically generatedc: 1110010(1)

Diagram

Description automatically generatedd:1000010(1)

Diagram

Description automatically generatedE:0110000(1)

Diagram, shape

Description automatically generatede:0010000(1)

Diagram

Description automatically generated F: 0111000(1)

Diagram

Description automatically generatedr : 1111010 (1)

Diagram

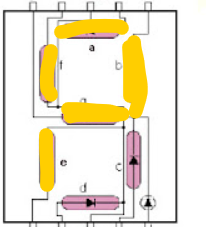
Description automatically generatedu : 1100011 (1)

Diagram

Description automatically generatedU: 100001(1)

Diagram

Description automatically generatedo: 1100010(1)

P: 0011000(1)

Diagram

Description automatically generatedh: 1101000(1)

Diagram

Description automatically generatedH: 1001000(1)

Diagram

Description automatically generatedn: 1101010(1)

Diagram

Description automatically generated l:1111001(1)

Diagram

Description automatically generated L: 1110001(1)

A picture containing scatter chart

Description automatically generated1001111

A picture containing shape

Description automatically generated0010010

A picture containing icon

Description automatically generated0000110

Icon

Description automatically generated1001100

A picture containing shape

Description automatically generated"0100100

A picture containing icon

Description automatically generated0100000

A picture containing icon

Description automatically generated0001111

Shape, icon

Description automatically generated0000000

Shape

Description automatically generated with low confidence0000100

Shape

Description automatically generated00000011

# 1.3.6 Afisare pe SSD-Schema bloc

Diagram

Description automatically generated

# 1.3.7. Codificare VHDLDiagram Description automatically generated with medium confidence

# 1.3.8 Afisare pe SSD- Codificare VHDL

Table

Description automatically generated

Pe input vom primi 16 biti, care reprezinta: tipul de tasta apasata(primii 8 biti), urmat de codificarea tastei.

HGFEDCBA: codificare valori

# 2. Implementare in VHDL

# 2.1 Implementare debouncer

Debouncer-ul numit si circuit monoimpuls, este folosit pentru butoanele care sunt apasate. Fiindca noi vom apasa taste, este posibil ca acestea sa fie uzate, fapt ce va determina un tren de impulsuri, cand, in realitate s-a apasat o singura data.

Debouncer-ul este alcatuit dintr-un numerator, care va fi implementat pe un numar cat mai mare de biti, in functie de uzura butonului, acesta reprezinta perioada de esantionare.

In lista de elememente care contribuie la realizarea circuitului, se Numara si un set de 3 bistabile, care propada un tren de impulsuri. Cand se ajunge la 11…1 pe numerator, se preia valoare butonului. Cand se ajunge la ultimul bistabil, se face o operatie de SI intre penultimul bit introdus, cu ultimul bit din bistabil negat.

Logica pentru SI, are drept scop activarea o singura data a circuitului, chiar daca se va mai detecta in viitor un 1. Aceasta valoare de 1, nu va face o noua activare a circuitukui.

Diagram

Description automatically generated

# 2.2 Implementare Afisare taste apasate

Pentru implementarea afisarii pe SSD, m-am folosit de faptul ca pe acesta se va afisa reprezentarea unei singure litere/cifre. Din acest motiv, vom folosi un singur Anod si Catozii pe care ii avem.

Ca input, vom peimi un semnal pe 16 biti, reprezentand cele 4 cifre care transmit codificarea tastei apasate. Primele 2 cifre, reprezinta tipul de tasta(normala/shift), iar a doua grupare de 8 biti, reprezinta codificarea propriu-zisa a tastei.Shape

Description automatically generated

Din punctul de vedere al al afisarii, am codificat bitii corespunzatori care vor fi plasati anozilor, pentru a se afisa litera dorita, pentru acest lucru, am folosit un case, urmat de codificarea corespunzatoare.

# 3. Controller: PS/2

2.3.1. Synchronize recive dataDiagram

Description automatically generated

Text

Description automatically generated

2.3.2. Datele primate prin PS/2 -pe front descrescator

Diagram

Description automatically generated

Text

Description automatically generated with medium confidence

2.3.3. Se verifica daca: bitul de paritate, stop, start sunt corecti

Diagram

Description automatically generated

Regula pentru validare:

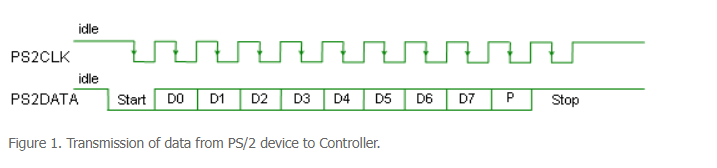
Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Transmisia datelor pe front descrescator



# 2.3.4. Generare intrerupere

Diagram

Description automatically generated

# 2.4.1. Controlarea tastelor apasate=>generarea codurilor ASCII

# 2.4.2. Retinerea tastelor apasate

Intrucat exista posibilittaea sa se apese simultan 2 taste, ex: Shift+A, trebuie sa avem doua variabile, in care sa se memeoreze valoarea anterioara(Shift)+ Tasta nou apasata(A)

Text

Description automatically generated

# 4.Legarura intre cod si componente:

**PS2- descries in** ps2\_keyboard.vhdl

A picture containing text

Description automatically generated

# Detalii implementare

Pentru implementare, vom folosi clk-ul placii si cel al ps2. Cel am placii este de 18\_000 de ori mai mare, din acest motiv pe fiecare ciclu de ceas al placii vom analiza ce comanda s-a transmis. Dupa 8 cicluri de ceas, vom avea noul cod de control, dupa ce se transmite scan\_code-ul, de pe clk-ul placii, se va analiiza/trensmite, pe clk-ul ceasului

Intrerupere: in mod implicit, la apasarea unui cCTRL, Shift, se genereaza coduri de intrerupere. Se foloseste in acest sens un fleg pentru a mentine informatiaa de break.

## 6.1. Ierarhia de fisiere

Graphical user interface, text, application, chat or text message, email

Description automatically generated

## 6.2. Receiver

Text

Description automatically generated

Are rolul de a captat cate un cod primit, verifica codurile, validabdu-se tipul de paritate, iar apoi transmite rezultatul obtinut, dupa formarea cuvantului de 11 biti.

Clk-ul placutei are o frecventa mai mare decat clk-ul ps2-ului, din acest motiv, trebuie sa folosim bistabile D, pentru a mentine valoarea corecta pana aceasta va fi citita de PS2. (frecventa ceas: 100\_000\_000, iar ps2=20\_000.

Pentru a determina paritatea si eroarea, se face un xor intre bitii adiacenti, mai precis: --bitul de start=0, bitul de stop=1,verificarea cu parity bit, avem paritate impara , altfel, eroare.

Se va putea trece la urmatorul bit citit, daca, s-a ajuns la 5\_000ns ⬄ 100\_000\_000/20\_000, altfel, se mentine citirea.

Dupa citire, se activeaza “enable”, care anunta citirea complete.

## 6.3. Transmiter

Text

Description automatically generated

In aceasta parte, se verifica ce taste au fost apasate, in functie de scan code-urile generate.

Am folosit in implementare un FSM, pentru a decide care in ce stare va intra automatul in cazul in care s-a primit/generat un cod correct.

Cand tastatura este conectata, se va transmite de la reciver, codul 0xFF, care semnifica faptul ca este in staraea de primire a semnalului. Cand Transmiter-ul va detecta codul 0xFF, se va transmite codul de activare “0xAA”, care semnifica faptul ca se pot receptiona coduri de la taste, respective combinatii de taste.

Dupa aceasta initializare, se vor putea accepta coduri pentru CapsLock si pentru Ctrl, Alt…

Cazuk in care se transmite codul:

“59”- s-a detectat un shift, vom active un flag, care va fi de ajutor la decodificari caractere.

“12” -shift….

In functie de fleg-uri determine codurile ascii, care sunt in documentatie, pentru fiecare caracter.

**Obs**: Eu am decodificat toate valorile, pentru o posibila implementare viitoare.

Daca nu se mai citeste nimic, atunci, se activeaza un flag: break, pentru a se decodifica valoarea.

Se va transmite la iesire, codul ascii corespunzator literei cititie.

## 6.4. Afisaj

Text

Description automatically generated

Vom afisa caracterele primate pe afisajul SSD, in functie de codul ASCII, primit pe intrarea Data.

1. Rezultate experimentale

A picture containing chart

Description automatically generated

Functionalitatea a fost testata pe placuta Nexys4.

Problemele aparute pe parcurs, au fost la generarea perioadei pe care se proceseaza tastele, astfel incat sa se citeasca tot scandode-ul.

La testarea finala, rezultatele au fost cele asteptate.

In cadrul laboratorului/proiectului, s-au testat toate cazurile implementate, se pot consulta imaginile de mai jos.

## 7.1. Sinulare-video

Pentru a observa functionalitatea in timp real, se poate urmari videoclipul realizat in timpul laboratorului:

<https://drive.google.com/file/d/16O4uqwVNLrvX-2C1hvPnq8CWIF6T9bAB/view?usp=sharing>

## 7.2 Etape testare

Am avut nevoie de o tastatura, pentru simulare, acest lucru a ridicat un set de probleme, intrucat doua dintre tastaturile pe care am testat proiectul nu transmiteau semnalul de ps2, iar pe afisaj nu se putea vedea nimic, fapt ce a condus catre o cerectare mai profunda a codului si modificari ale acestuia in cazul in care nu era nevoie.

Proiectul se comporta condorm asteptarilor, dar doar daca se foloseste o anumita atastatura, de pilda, cele pe care le detin eu, nu se potrivesc cu placuta, doar pe cea din laborator am pututt face simularea, in care se observa rezultatele asteptate.

1. .Imbunatatiri ulterioare

Pe viitor s-ar putea analiza apasarea tastelor F, a combinatiilor de taste si a tastelor pentru tastatura numerica.

De asemenea s-ar putea implementa logica pentru caracterele de sub tastele F, de exemplu, semnele de punctuatie, @, %, & si altele.

S-ar putea implementa afisarea tuturor caracterelor citite, in ordinea in care au fost procesate. Pentru interactivitate, s-ar putea implementa logica pentru sageti, care sa modifice pozitia unui caracter afisat.

# Concluzii

Pentru implementare a fost necesara citirea documentatiei si transpunerea in cod VHDL.

In urma documentarii, implementarii si testarii, am obtinut rezultatul asteptat.

Proiectul respecta toate cerintele mentionate si are o functionalitate corecta, putandu-se vedea in materialul de mai sus imaginile din timpul testatii.

De remarcat este faptul ca nu s-au putut afisa toate literele mari, din acest motiv, se va afisa 0 pe SSD daca nu exista reprezentare.

Limitarea este data de numarul de afisaje digitale. Pentru o implementare viitoare, care ar pune la dispozitie un SSD mai generos, codul poate identifica apasrea oricarei litere sau combinatii, iar in viitor se pot modifica liniiile de cod ale reprezentariii pe Anozi.

# Bibliografia:

1. <https://digilent.com/reference/learn/programmable-logic/tutorials/nexys-4-ddr-keyboard-demo/start>
2. <https://www.academia.edu/32460779/Nexys4_DDR_FPGA_Board_Reference_Manual>
3. <https://www.win.tue.nl/~aeb/linux/kbd/scancodes-10.html#keyboardid>
4. <https://techdocs.altium.com/display/FPGA/PS2_W+-+Wishbone+PS2+Controller>
5. <https://techdocs.altium.com/display/FPGA/PS2_W+-+Transmission+Protocols>
6. <https://techdocs.altium.com/display/FPGA/PS2+Commands>
7. <https://www.fpgakey.com/tutorial/section924>
8. <https://wiki.osdev.org/PS/2_Keyboard>