

**FACULTATEA: Automatică și Calculatoare  
SPECIALIZAREA: Calculatoare și Tehnologia Informației  
DISCIPLINA: Digital Systems Design  
PROIECT: Controller de tastatură PS/2**

PROFESOR ÎNDRUMĂTOR : PROIECT REALIZAT DE:

Bar Luca-Narcis

Vlad Cristian Miclea Cornea Mihai

Grupa 30411

**Cuprins**

**1.Specificatia proiectului.................................................................................................3**

**2.Schema bloc..................................................................................................................8**

**3.Proiectarea....................................................................................................................9**

**4.Justificarea solutiei alese și posibilități de dezvoltare..............................................19**

**5.Instrucțiuni de utilizare..............................................................................................20**

**6.Bibliografie...................................................................................................................21**

**Capitolul 1. Specificația proiectului**

**Cerința**

*„Să se proiecteze un controller de tastatură PS2. Se cere citirea tastelor şi afişarea de caractere corespunzătoare pe afişajul cu 7 segmente. Se vor vedea ultimele 4 simboluri, iar tastele de control vor avea roluri speciale (ex.: Enter începe un rând nou, ştergând afişajul). Poziţia punctului pe afişor va fi controlată cu ajutorul săgeţilor. Documentaţie: manualele de referinţă pentru plăcile cu FPGA şi documentaţia pentru protocolul PS2* <file:///C:/Users/user/AppData/Local/Temp/Rar$DIb4740.34663/PS2Protocol.pdf>*. Proiectul va fi realizat de 2 studenţi.”*

**Despre Proiect**

În primul rând, este important să înțelegem cum se realizează recepția datelor de la tastatura PS/2.Tastatura și mouse-ul au fost folosite încă de la începutul fabricării sistemelor informatice. Cu toate că noile modele au apărut pe piațăîn mod constant, principiul de bază al funcționării lor a rămas același. Protocolul recunoaște caracterul citit ca un cod hexazecimal trimis de la tastatură . Conexiunea poate fi făcută prin diferite tipuri de porturi , cum ar fi PS - 2 ( Personal System 2 ) pus în aplicare de către IBM , USB ( Universal Serial Bus ) , ADB (Apple Desktop Bus ) puse în aplicare de către Apple .

În cazul nostru , tastatura se conectează la calculator printr-o tehnologie numită PS/2 . Scopul nostru principal este de a realiza această conexiune cu DIGILENT Nexys 2, utilizând limbajul de proiectare hardware VHDL și programele Active HDL și ISE Design Suite 14.7 . La finalul implementării, proiectul îi va permite utilizatorului să afișeze pe afișajul BCD 7 segmente al plăcuței Nexys 2 caracterele transmise de la tastatură, să golească afișorul prin simpla apăsare a tastei ENTER, să activeze și să modifice poziția punctului prin apăsarea săgeților și nu în ultimul rând să golească câte un afișor pe rând cu ajutorul tastei BACKSPACE.

**Protocolul PS/2**

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

Vom prezenta diagrama pinilor conectorului PS/2.

|  |  |  |  |
| --- | --- | --- | --- |
| The PS2 connector | 1. KBD Clock 2. GND 3. KBD Data 4. N/C 5. +5V (VCC) 6. N/C | The PS2 5pins connector | 1. KBD Clock 2. KBD Data 3. N/C 4. GND 5. +5V (VCC) |

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 . Fiecare bit este trimis în serie cu următoarea ordine :

= Bitul de start > 0 ... 7 biți de date = > bit de paritate = Impar > bit de stop . ( 8 / O / 1 )

Datele sunt transmise începând cu bitul cel mai puțin semnificativ ( LSB ), bitul cel mai semnificativ ( MSB ) fiind ultimul.

Ceea ce se transmite sunt scancode-urile caracterelor . De exemplu, tasta ' i' are scancode-ul 0x43 care se traduce 0100 0011 în binar , dar se transmite începând cu bitul cel mai puțin semnificativ și devine 1100 0010. Urmează să fie adăugată paritatea ( aici este 0) și bitul de start ( 0 ) și cel de stop ( 1 ), ceea ce înseamnă că la apăsare se transmite ' 0 1100 0010 0 1 '.

Fiecare bit este citit pe frontul descrescător al ceasului pentru a fi păstrați corect sincronizați:



La apăsarea unei taste, tastatura va trimite codul de scanare al tastei pe care ați apăsat la gazdă. Tastatura distinge două stări pentru cheie: starea down (apăsare) și starea up(release). Tastatura va trimite codul de scanare pentru cheia pe care ați apăsat (starea down) și atunci când lăsați cheia în sus va trimite un "F0", urmată de codul de scanare al cheii (starea up).

De exemplu, dacă apăsați tasta "A", codul de scanare 1C va fi trimis. Când cheia va fi lăsată, F0 1C va fi trimis. Dacă apăsați pe tasta "A" și mențineți apăsat mai mult decât este typematic delay-ul, tastatura va continua să trimită codul de scanare a cheii în funcție de typematic rate, până când se eliberează tasta.

Toate datele sunt transmise cate un byte pe rând și fiecare byte este trimis într-un cadru format din 11-12 biți. Acești biți sunt:

• 1 bit de start. Aceasta este întotdeauna 0.

• 8 biți de date, bitul cel mai puțin semnificativ mai întâi.

• 1 bit de paritate (impar).

• 1 bit de stop. Aceasta este întotdeauna 1.

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 corectitudinii datelor din pachetul curent și este setat la 1 în cazul în care în pachet se află un număr par de biți high, sau la 0 în cazul în care în pachet există un număr impar de biți high. La apăsarea unei taste se generează un *”make code”* care este unic pentru fiecare tastă. În momentul în care aceasta este eliberată se transmite un *”break code”*. Un *”break code”* este format din codul *”F0”* urmat de *”make code”*-ul corespunzător tastei. Dacă acesta din urmă este format din doi octeți atunci codul *”F0”* este plasat între acești doi octeți. Modul în care calculatorul interpretează codul primit ține de felul în care acesta a fost programat. Majoritatea calculatoarelor lasă aceasta sarcină plăcii de bază, care transformă*”scan-code”*-ul primit de la tastatură într-un cod ASCII. Valorile codurilor ASCII se găsesc în imaginea de mai jos.



**Tastatura în hexazecimal**

Codurile Hexazecimale corespunzătoare tastelor pentru care este implementat proiectul:

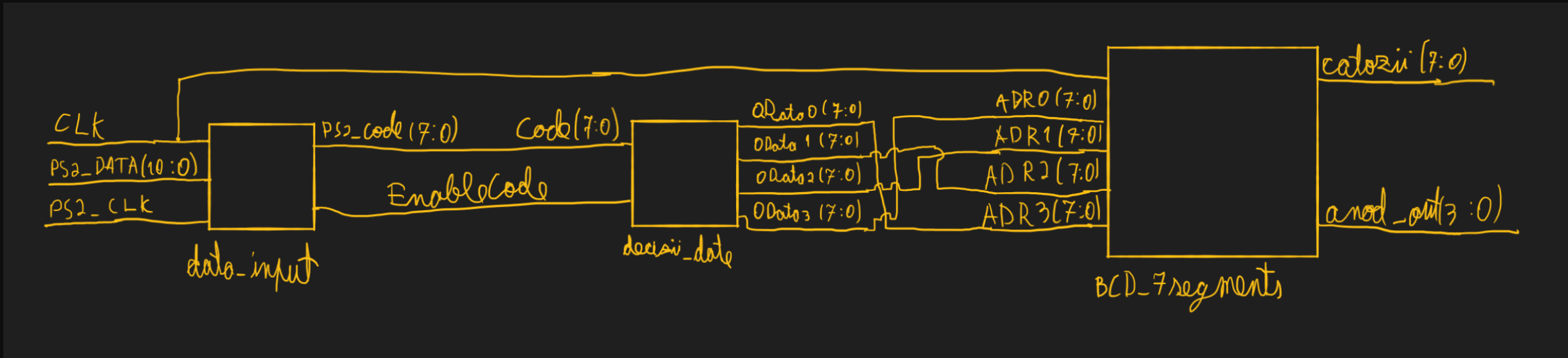




**Limbajul VHDL**

**Pentru implementare,am folosit limbajul descriptiv VHDL și programul Active-HDL oferit de Aldec.**

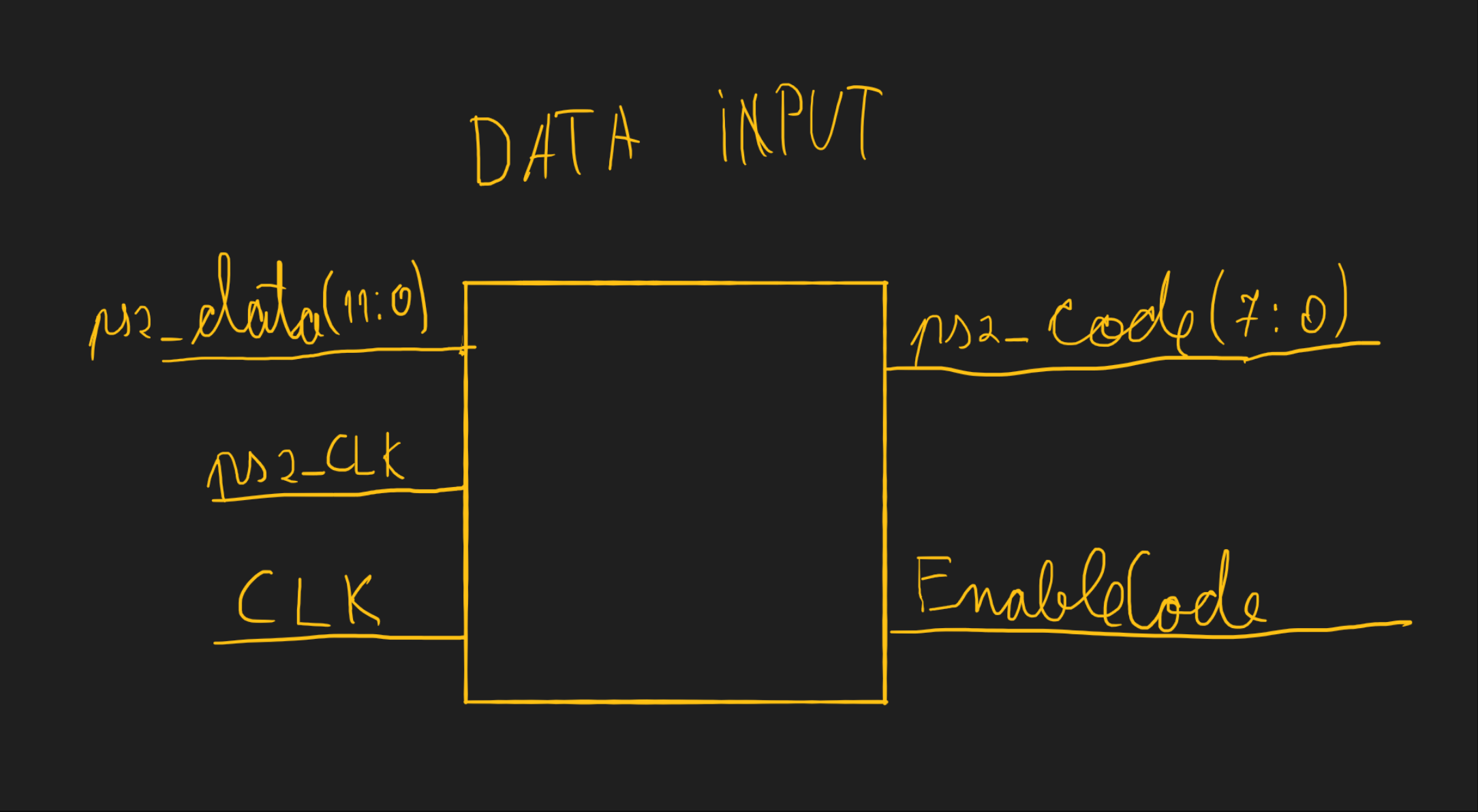
**Capitolul 2. Schema Bloc**



**Capitolul 3. Proiectarea**

Pentru a reuși sa descriem proiectul in limbaj VHDL, am decis sa impartim codul in 4 entitati, care vor fi descrise separat, în detaliu, în cele ce urmează.

**Unitatea de primire a datelor de la tastatură:data\_input**



Această unitate este specializată în primirea serială a celor 11 biți de la tastatură pe frontul descendent a clock-ului tastaturii, pe lângă aceste intrări mai are și clock-ul plăcuței. Ca și output-uri avem: un semnal new\_code, un flag, care reprezintă un ENABLE și care pentru valoarea ”1” anunță că datele au fost preluate corect și sunt pregatite să fie decodificate de decizii\_date, apoi avem un vector de 8 biti, ps2\_code, care semnifică reprezentarea în hexazecimal a unei taste.

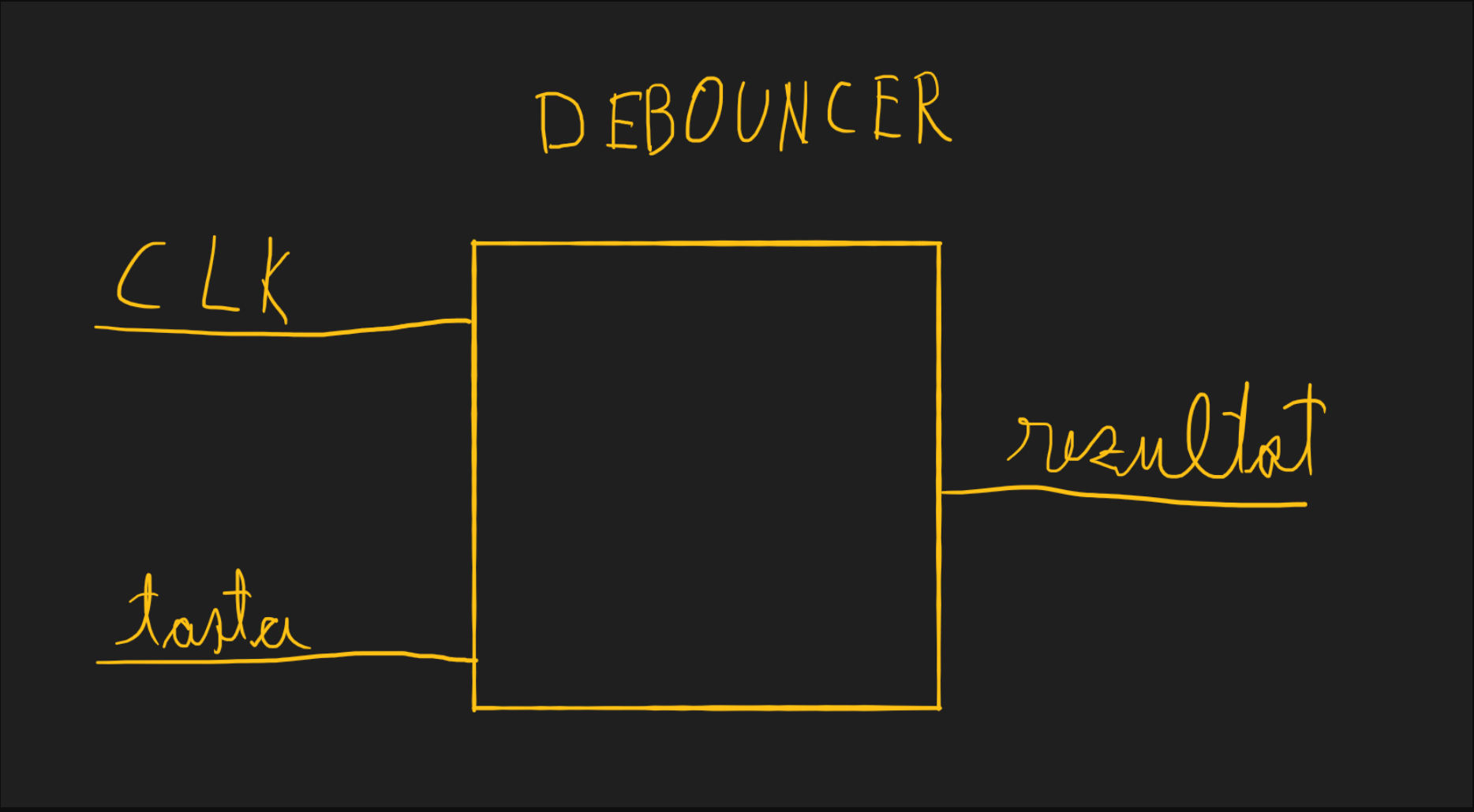
Pentru a verifica daca secventa de biți este corectă, am folosit următoarea secventa de cod:

eroare<=NOT(NOT vector11(0) AND vector11(10) AND (vector11(9) XOR vector11(8) XORvector11(7) XOR vector11(6) XOR vector11(5) XOR vector11(4) XOR vector11(3) XOR vector11(2) XOR vector11(1)));

În urma acestei instrucţiuni, eroare va lua valoarea ’1’ dacă secvenţa de biţi nu corespunde condiţiilor. În caz contrar,eroare va lua valoarea ’0’.

Mai avem prezent și un count\_idle,care asigură faptul ca datele primite vor fi transmise mai departe doar după ce au fost introduși toți cei 11 biți.

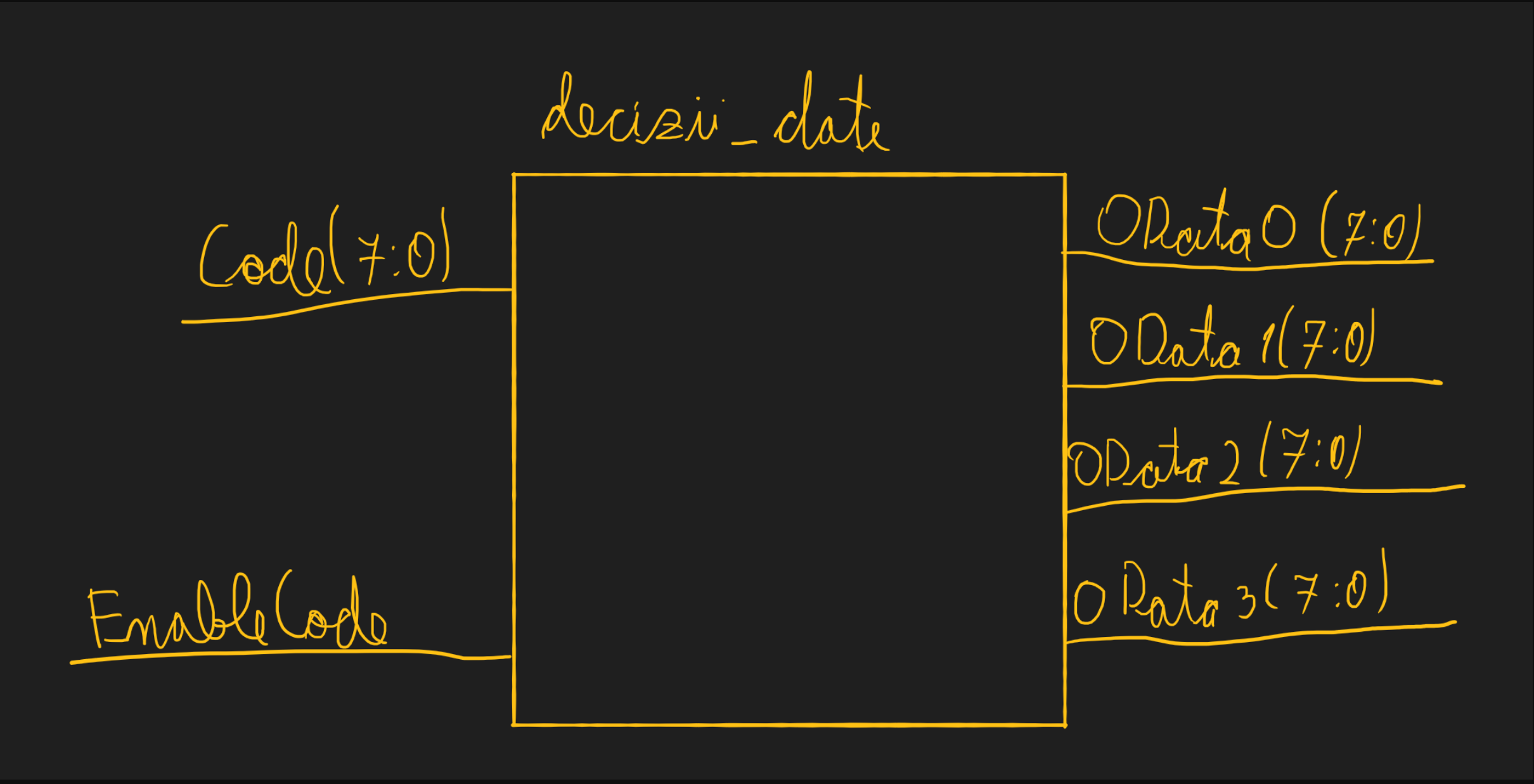
De asemenea, în arhitectura acestei unitati avem si un circuit DEBOUNCER, cu rolul de a stabiliza primirea bitilor de la aceasta. Aceasta stabilizare este aplicata atât pentru ps2\_clock, cât și pentru ps2\_data.(Menționăm că această componentă este utilă în cazul implementării pe plăcuță,deoarece în cazul simulării în Active-HDL,nu apare problema stabilizării).



Semnificatia notatiilor :

* CLK - tact de ceas primit de la placă
* ps2\_clk - tact de ceas primit de la tastatură
* ps2\_data - intrarea serială a biților primiți de la tastatură
* EnableCode - un enable care are valoarea '1' daca datele sunt corecte
* Count\_idle-numără la fiecare rising edge al clock-ului,mai precis numără în momentul în care este introdus un bit prin ps2\_data
* ps2\_code(7 downto 0) - cei 8 biţi informaționali care reprezintă codificarea hexazecimală a unei taste

**Unitatea de decodificare si interpretare:decizii\_date**



Această unitate primeşte ca input-uri EnableCode ( enable-ul generat de componenta precedentă ) şi Code ( reprezentat de ps2\_code generat în componenta de data\_inputs). Output-urile sunt reprezentate de 4 vectori a câte 8 biţi OData0, OData1, OData2, OData3, fiecare având reprezentarea în cod hexazecimal a poziţiei din memorie (adresa) la care se află reprezentarea tastelor pe BCD 7-Segmente. Aceşti vectori corespund anozilor de pe afişor de la dreapta la stânga.

În interiorul aceste unități va fi recepționat codul tastelor primite,și va fi realizată interpretarea sa:

-în cazul în care este recpționat codul unei taste normale(de ex. litere/cifre),acesta va fi transmis și afișat pe outputul corespunzător

-în cazul în care este recepționat codul tastei ENTER,toate afișoarele se golesc,primind codul lui ENTER

-în cazul lui BACKSPACE,se golește vectorul corespunzător,și este introdus codul tastei BACKSPACE

-în cazul săgeților(stânga sau dreapta) se face deplasarea punctului între vectori(afișori).

Următoarea porţiune de cod realizează funcţia tastei ENTER:

If Code = "01011010" then --Daca a fost apasata tasta Enter, se goleste afisorul

Data3<="01011010";

Data2<="01011010";

Data1<="01011010";

Data0<="01011010";

backsp\_pos <= 0;

pos<=0;

else .....

Data0, Data1, Data2, Data3 reprezintă semnale auxiliare din interiorul arhitecturii acestei componente, pe care le folosim în proces pentru a modifica vectorii.

Semnalele **pos** şi **backsp\_pos** sunt folosite pentru a reţine poziţia punctului, respectiv poziţia pe care ne aflăm în prezent, în cazul în care Backspace-ul a fost apăsat.

Secvenţă de cod următoare implementează mutarea punctului cu ajutorul săgeţii drepte:

if (falling\_edge(EnableCode)) then --valid doar pe front descendent

if tData1="11100000" and Code = "01110100" then --Sageata dreaptaE0 74

case pos is

when 0 => --start

Data0(7) <= '1';

Data1(7) <= '0';

Data2(7) <= '0';

Data3(7) <= '0';

pos <= 1;

backsp\_pos <=1;

when 1 =>

Data0(7) <= '0';

Data1(7) <= '0';

Data2(7) <= '0';

Data3(7) <= '1';

pos <= 4;

backsp\_pos <=4;

when 2 =>

Data0(7) <= '1';

Data1(7) <= '0';

Data2(7) <= '0';

Data3(7) <= '0';

pos <=1;

backsp\_pos <=1;

when 3 =>

Data0(7) <= '0';

Data1(7) <= '1';

Data2(7) <= '0';

Data3(7) <= '0';

pos <=2;

backsp\_pos <=2;

when 4 =>

Data0(7) <= '0';

Data1(7) <= '0';

Data2(7) <= '1';

Data3(7) <= '0';

pos<=3;

backsp\_pos <=3;

when others=>null;

end case;

Codul este similar şi pentru săgeata stânga.

Datorită faptului că săgeţile sunt taste speciale, al căror cod se scrie pe 2 bytes, trebuie să avem în considerare comportamentul diferit al tastaturii la apăsarea acestora. Dintre cei doi bytes, ştim că primul trebuie să fie xE0, aşa că ne folosim de tData1 pentru a reţine codul precedent perceput de către sistem. În cazul în care acesta a fost într-adevăr xE0, ştim că primim semnal de la taste corespunzătoare săgeţilor, aşa că verificăm dacă al doilea byte corespunde make code-urilor corespunzătoare acestora: x6B pentru stânga, respectiv x74 pentru dreapta.

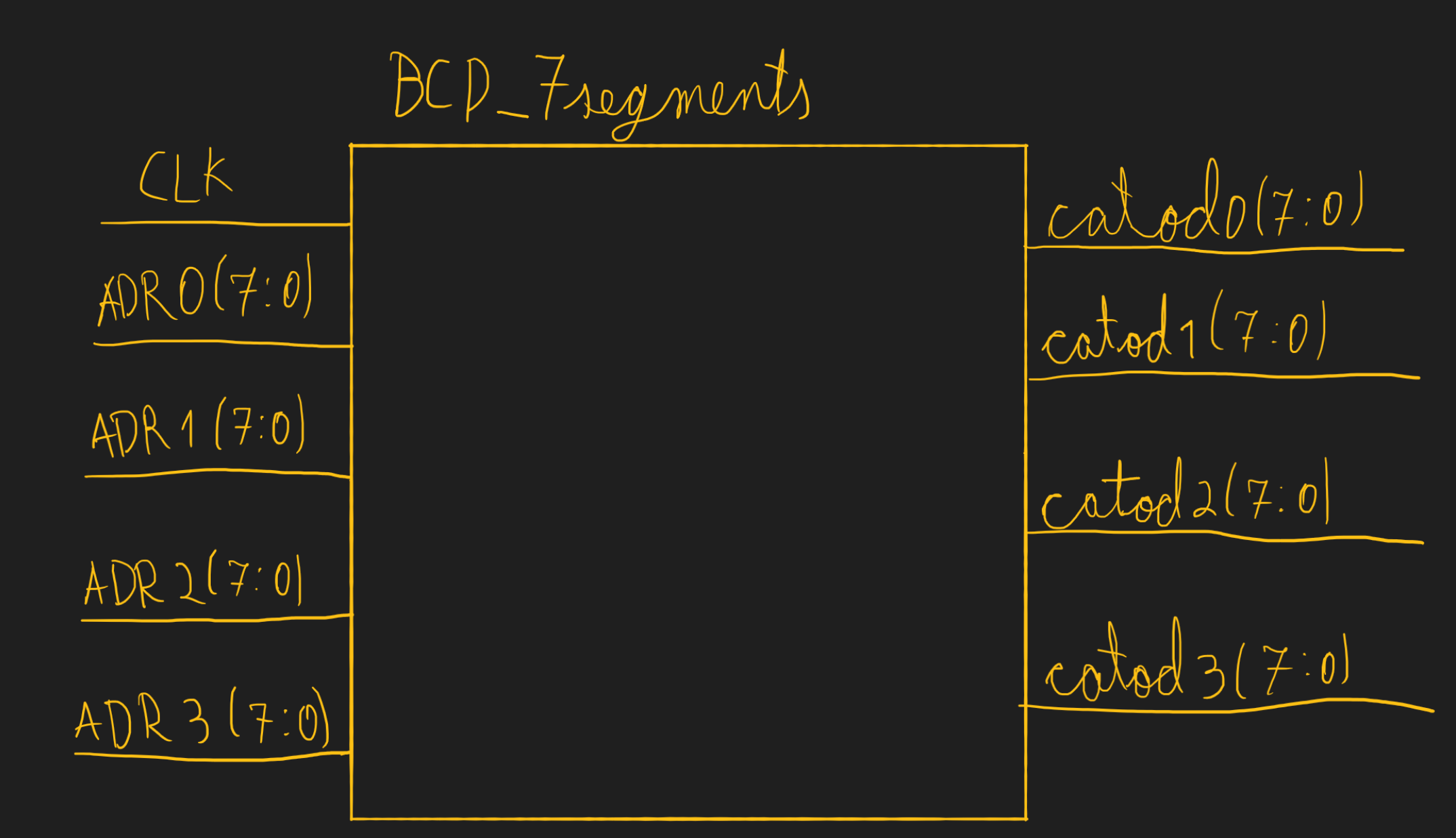
Semnificatia notatiilor:

* EnableCode- enable generat anterior
* Code(7 downto 0) - vectorul din ps2\_code
* OData0,OData1,OData2,OData3( 7 downto 0) – vectorii care contin informatia(adresele)

! De menționat,pentru introducerea fiecărei taste,trebuie introdus codul sau în hexa,mai apoi tasta intermediară care semnifică apăsarea sa(ex. pt săgeți E0,iar pt restul F0),urmând mai apoi din nou codul tastei.

Exemplu,pt tasta A se va introduce 1C,mai apoi F0,și la final din nou 1C,pentru o reprezentare corectă.

**Afișorul: BCD\_7segments**



Pentru această unitate, input-urile sunt clock-ul CLK şi cei 4 vectori a câte 8 biţi preluaţi de la unitatea decizii\_date pe care trebuie să îi descifrăm.

În această componentă, cu ajutorul unei memorii cu 256 de locaţii (0 to 255) am realizat corespondenţa dintre byte-ul perceput de la tastatură şi reprezentarea tastelor pe afişorul BCD 7 Segmente. Am observat că reprezentările în codul decimal al tastelor corespunzătoare cifrelor şi literelor nu pot fi mai mari de 127 ( mai precis, MSB al vectorilor este întotdeauna ’0’) şi ne-am folosit de acest detaliu pentru a coda stările în care punctul se află pe afişor împreună cu caracterul corespunzător byte-ului. Astfel, reprezentarea fiecărui caracter pe care dorim să îl afişăm se află la poziţia din memorie corespunzătoare reprezentării în decimal al make code-ului transmis de tastatură.

Exemplu: tasta ' A' are make code-ul 1C, adica 00011100 ,în decimal 28. Activarea punctului pe anodul corespunzator tastei ' A' se face prin setarea bitului cel mai semnificativ => 10011100,in decimal, 156.

În cadrul memoriei este făcută corespondența dintre codul decimal al tastelor și reprezentarea acestora pe afișorul cu 7 segmente.

Codificarea cifrelor este urmatoarea:

--CIFRE

69 => "11000000" , --0 Fără.

197 => "01000000" , --0 Cu .

22 => "11111001" , --1 Fără.

150 => "01111001" , --1 Cu .

30 => "10100100" , --2 Fără.

158 => "00100100" , --2 Cu .

38 => "10110000" , --3 Fără.

166 => "00110000" , --3 Cu .

37 => "10011001" , --4 Fără.

165 => "00011001" , --4 Cu .

46 => "10010010" , --5 Fără.

174 => "00010010" , --5 Cu .

54 => "10000010" , --6 Fără .

182 => "00000010" , --6 Cu .

61 => "11111000" , --7 Fără .

189 => "01111000" , --7 Cu .

62 => "10000000" , --8 Fără .

190 => "00000000" , --8 Cu .

70 => "10010000" , --9 Fără .

198 => "00010000" , --9 Cu .

Codificarea literelor:

28 => "10001000" ,--A fara .

156 => "00001000" ,--A cu .

50 => "10000011" ,--B fara .

178 => "00000011" ,--B cu .

33 => "11000110" ,--C fara .

161 => "01000110" ,--C cu .

35 => "10100001" ,--D fara .

163 => "00100001" ,--D cu .

36 => "10000110" ,--E fara .

164 => "00000110" ,--E cu .

43 => "10001110" ,--F fara .

171 => "00001110" ,--F cu .

52 => "11000010" ,--G fara .

180 => "01000010" ,--G cu .

51 => "10001001" ,--H fara .

179 => "00001001" ,--H cu .

67 => "11001111" ,--I fara .

195 => "01001111" ,--I cu .

59 => "11100001" ,--J fara .

187 => "01100001" ,--J cu .

66 => "10001011" ,--K fara .

194 => "00001011" ,--K cu .

75 => "11000111" ,--L fara .

203 => "01000111" ,--L cu .

58 => "11101010" ,--M fara .

186 => "01101010" ,--M cu .

49 => "10101011" ,--N fara .

177 => "00101011" ,--N cu .

68 => "10100011" ,--O fara .

196 => "00100011" ,--O cu .

77 => "10001100" ,--P fara .

205 => "00001100" ,--P cu .

21 => "10011000" ,--Q fara .

149 => "00011000" ,--Q cu .

45 => "10101111" ,--R fara .

173 => "00101111" ,--R cu .

27 => "10010011" ,--S fara .

155 => "00010011" ,--S cu .

44 => "10000111" ,--T fara .

172 => "00000111" ,--T cu .

60 => "11000001" ,--U fara .

188 => "01000001" ,--U cu .

42 => "11100011" ,--V fara .

170 => "01100011" ,--V cu .

29 => "11010101" ,--W fara .

157 => "01010101" ,--W cu .

34 => "10011011" ,--X fara .

162 => "00011011" ,--X cu .

53 => "10010001" ,--Y fara .

181 => "00010001" ,--Y cu .

26 => "10100101" ,--Z fara .

154 => "00100101" ,--Z cu .

Semnificație notații:

* ADR0,ADR1, ADR2,ADR3 ( 7 downto 0)- vectori de informaţie care intră în memorie
* catod0,catod1,catod2,catod3 ( 7 downto 0) - coduri pentru afișarea caracterelor primite pe afișorul 7 segmente

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 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 |

**Capitolul 4. Justificarea soluţiei alese și posibilități de dezvoltare**

Am ales această soluție deoarece părea cea mai intuitivă metodă de a realiza protocolul acestei tastaturi și corespunde protocolului descris la începutul documentului.Codul scris este destul de ușor de înțeles și este destul de ușor de modificat pentru implementarea pe o plăcuță FPGA(ex.NEXYS2)

Ca și posibilități de dezvoltarea, putem menționa găsirea unei soluții pentru reprezentarea incorectă a săgeților în cadrul simulării fișierului ps2\_controller(fișierul "top"). Simularea lor funcționează în cadrul fișierului decizii\_date,dar în fișierul mare apare o problemă de sincronizare,deoarece tData se schimbă prea repede din E0 și nu apucă sa detecteze faptul că ps2\_code are codul săgeților. De asemenea se poate reprezenta cazul în care tasta apăsată este SPACEBAR.

**Capitolul 5. Instrucţiuni de utilizare**

În cadrul simulării,va fi selectat fișierul ps2\_controller și va fi inițializată simularea. Se vor adăuga pe waveform toate semnalele prezente,și de asemenea vor fi adăugate și vector11 și count\_idle,pentru o vizualizare mai ușoară a proceselor.

Pe clk si ps2\_clk vor fi puși stimulatori de clock de 100de mhz,iar pe ps2\_data se pot folosi atât formule cât și un hotkey pentru introducerea datelor.(Atenție! Trebuie introduse bitul de start(0),mai apoi cei 8 biti ai tastei,apoi bitul pentru paritate și bitul de stop(1),și de asemenea trebuie introduse sub forma:cod tastă-cod F0/E0-cod tastă).De menționat,trebuie introdus un '0' după bitul de stop de fiecare dată pentru a continua în mod corect introducerea tastelor. În folderul proiectului se pot găsi screenshot-uri cu simulări efectuate în mod corect. (Menționăm că pentru o mai bună înțelegere a modului de funcționare a proiectului,se recomandă simularea fiecărei componente,în afara debouncer-ului)

**Capitolul 6. Bibliografie**

* <http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/ps2/ps2.html>
* <https://www.digikey.com/eewiki/pages/viewpage.action?pageId=28278929>
* <https://www.youtube.com/watch?v=TM9Q5CKZnBU>
* <https://www.slideshare.net/n380/ps2-keyboard-vhdl-design-tutorial>