Universitatea Tehnică Cluj-Napoca

Facultatea de Automatică și Calculatoare

Secția Calculatoare

An 2021-2022 semestrul II

**

PROIECTAREA SISTEMELOR NUMERICE

Proiect: Aplicație pentru mouse

Profesor îndrumător: Lișman Dragoș Florin

Studenți: Strugar Mădălina-Alexandra

Pavel Mădălina-Denisa

Grupa 30213

Cuprins

1. Specificație proiect
2. Descriere schemă bloc cu componente

3. Proiectare și implementare

3.1. Proiectare componente, codul comentat pentru componente

3.2. Proiectare ansamblu, codul comentat pentru ansamblu

4. Lista de componente utilizate

5. Specificația notațiilor I/O și a semnalelor interne

7. Utilizare și rezultate

7.1 Resurse necesare

7.2 Descrierea utilizării

7.3 Rezultate obținute în modulul de simulare

8. Posibilități de dezvoltare

9. Bibliografie

Specificație proiect

Descriere: Implementați o aplicație care permite utilizatorului să contorizeze numărul de click-uri al unui mouse.

Cerințe de funcționare:

1. Existența unui buton **RESET** care va reinițializa contorul cu 0;
2. Statusul curent este afișat pe SSD(aici în modulul de simulare);
3. Apăsând pe **BUTON STÂNGA**, contorul va crește;
4. Apăsând pe **BUTON DREAPTA**, contorul va scădea;
5. Un buton /switch **REVERSE** este folosit pentru a inversa rolurile celor două butoane;
6. Un led **IS\_LEFT** se va aprinde când suntem în modul reverse.

Proiectul va fi realizat de **2 studenți.**

Descriere schema bloc cu componente

Cutia neagră



Schema bloc

Diagram

Description automatically generated

Proiectare si implementare

Organigrama

Diagram

Description automatically generated

Proiectare componente, codul comentat pentru componente

UC

Componenta UC are rolul de a implementa starile automatului in functie de intrari(clk, reverse, reset, buton\_st, buton\_dr).

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity uc is

Port ( clk : in STD\_LOGIC;

reset : in STD\_LOGIC;

reverse : in STD\_LOGIC;

buton\_st : in STD\_LOGIC;

buton\_dr : in STD\_LOGIC;

increase : out STD\_LOGIC;

decrease : out STD\_LOGIC;

is\_left : out STD\_LOGIC);

end uc;

architecture uc\_arch of uc is

type stare\_t is(st1,st2,st3,st4,st5,st6,st7);

signal stare,nxstare:stare\_t;

begin

actualizeaza\_stare:process(reset,clk)

begin

if(reset='1')then stare<=st1;

elsif(rising\_edge(clk))then stare<=nxstare;

end if;

end process actualizeaza\_stare;

tranzitii:process(stare,buton\_st,buton\_dr,reverse)

begin

increase<='0';

decrease<='0';

case stare is

when st1=>

if(reverse='1')then nxstare<=st3;

else nxstare<=st2;

end if;

when st2=>

if(buton\_st='1')then nxstare<=st5;

elsif(buton\_dr='1')then nxstare<=st4;

else nxstare<=st1;

end if;

when st3=>

if(buton\_st='1')then nxstare<=st6;

elsif(buton\_dr='1')then nxstare<=st7;

else nxstare<=st1;

end if;

when st4=>decrease<='1';increase<='0';

nxstare<=st1;

when st5=>decrease<='0';increase<='1';

nxstare<=st1;

when st6=>decrease<='1';increase<='0';

nxstare<=st1;

when st7=>decrease<='0';increase<='1';

nxstare<=st1;

end case;

end process;

led: process(reverse,reset)

begin

if(reset='1') then is\_left<='0';

elsif(reverse='1') then is\_left<='1';

else is\_left<='0';

end if;

end process;

end uc\_arch;

Numărător

Componenta **numarator** are rol dublu: de a numara incrementa/decrementa contorul si de a reseta valoarea acestuia

Aceasta componenta are ca intrari principale **clk** (cu frecventa de 10 nanosecunde), **reset** (cand are valoarea `1`, numaratorul este adus în starea 0 si semnalul **cnt** devine x”0000”), **increase** (un indicator de directie - daca este `1` numaratorul creste contorul), **decrease(**un alt indicator de directie - daca este `1` numaratorul descreste contorul). Iesirea componentei este un vector de 16 biți care reprezintă valoarea contorului.

**Numărător** are 4 mari funcții:

Cand **reset**=`1` numaratorul este adus starea “0”. Toate celelalte trei regimuri de functionare sunt atinse doar daca semnalul de reset este 0 logic.

Numaratorul numara crescator pe frontul ascendent al **clk**-ului daca **increase**=`1` si **decrease**=`0`, daca este atinsa starea “1111101000” la urmatorul impuls de tact numaratorul ajunge in starea “0”.

Numaratorul numara descrescator pe frontul ascendent al **clk**-ului daca semnalul **increase**=`0`si **decrease**=`1` și cnt>=”0000”.

Al patrulea regim de functionare a componentei **Numărător** reprezinta resetarea contorului la 0 cand este atinsa valoarea 1000.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity numarator is

Port ( increase : in STD\_LOGIC;

decrease : in STD\_LOGIC;

reset : in STD\_LOGIC;

clk : in STD\_LOGIC;

solutie : out STD\_LOGIC\_VECTOR (15 downto 0));

end numarator;

architecture arch of numarator is

signal cnt: STD\_LOGIC\_VECTOR(15 downto 0):=x"0000";

begin

process(clk,reset,increase,decrease)

begin

if(reset='1')then cnt<=x"0000";

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

if(increase='1' and decrease='0') then cnt<=cnt+1;

elsif(increase='0' and decrease='1' and cnt>"0000") then cnt<=cnt-1;

elsif(cnt="1111101000") then cnt<=x"0000";

end if;

end if;

end process;

solutie<=cnt;

end arch;

Display

Componenta **Display** are rolul de a afisa pe display-ul placutei FPGA informatia primita de la vectorul sol. Acest lucru este posibil cu ajutorul celor 7 catozi si cu celor 4 anozi, ambii fiind activi pe 0. Aceste semnale ale ecranului sunt de fapt iesirile blocului **Display**.

Introducerea informatiei in componenta se face printr-un semnal de tip vector numit **intrare\_numărător**, aflat în intervalul [0,1000].

In cadrul componentei **Display** semnalele de intrare sunt „impartite” in patru semnale de tip vector(fiecare de câte 4 biți), reprezentând câte o cifră(cifra unităților este reprezentată de vectorul **cifra\_0**(intrare\_numărător(3 downto 0)), cifra zecilor este reprezentată de vectorul **cifra\_1**(intrare\_numărător(7 downto 4)), cifra sutelor este reprezentată de vectorul **cifra\_2**(intrare\_numărător(11 downto 8)), cifra miilor este reprezentată de vectorul **cifra\_3**(intrare\_numărător(15 downto 12))) . Este folosit un numarator si doua multiplexoare, cu ajutorul carora alegem ce informatie sa se afiseze cu ajutorul catozilor si ce anod sa fie activ la un moment dat.

Numaratorul foloseste un clock cu perioada de 10 ns. Cei mai semnificativi biți(15,14) indica activarea unui anumit anod si aparitia pe ecran/simulare a unei valori corespunzatoare:

Daca numaratorul este in starea 1, vectorul ieșire\_anod primeste valoarea „1110”, semnalul care indica valoarea de afisat primeste valoarea semnalului **cifra\_0**.

Daca numaratorul este in starea 1, vectorul ieșire\_anod primeste valoarea „1101”, semnalul care indica valoarea de afisat primeste valoarea semnalului **cifra\_1**.

Daca numaratorul este in starea 1, vectorul ieșire\_anod primeste valoarea „1011”, semnalul care indica valoarea de afisat primeste valoarea semnalului **cifra\_2.**

Daca numaratorul este in starea 1, vectorul ieșire\_anod primeste valoarea „0111”, semnalul care indica valoarea de afisat primeste valoarea semnalului **cifra\_3**.

Aceste asignari sunt realizate cu primul multiplexor.

Mai apoi valoarea semnalului **ieșire\_mux1** este tradusa intr-o secventa de 7 biti reprezentata pe afisor (al doilea multiplexor).

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity display is

Port ( clk : in STD\_LOGIC;

intrare\_numarator : in STD\_LOGIC\_VECTOR (15 downto 0);

anod : out STD\_LOGIC\_VECTOR (3 downto 0);

catod : out STD\_LOGIC\_VECTOR (6 downto 0));

end display;

architecture arth of display is

component mux4\_1 is

port(intrare0: in STD\_LOGIC\_VECTOR(3 downto 0);

intrare1: in STD\_LOGIC\_VECTOR(3 downto 0);

intrare2: in STD\_LOGIC\_VECTOR(3 downto 0);

intrare3: in STD\_LOGIC\_VECTOR(3 downto 0);

s:in STD\_LOGIC\_VECTOR(1 downto 0);

iesire: out STD\_LOGIC\_VECTOR(3 downto 0));

end component;

component bcd\_7\_seg is

port(BCD\_in: in STD\_LOGIC\_VECTOR(3 downto 0);

BCD\_out: out STD\_LOGIC\_VECTOR(6 downto 0));

end component;

signal cifra\_0,cifra\_1,cifra\_2,cifra\_3,anod\_0,anod\_1,anod\_2,anod\_3,iesire\_mux1,iesire\_anod: std\_logic\_vector(3 downto 0);

signal iesire\_catod: std\_logic\_vector(6 downto 0);

signal selectie: STD\_LOGIC\_VECTOR(1 downto 0);

signal counter\_display: std\_logic\_vector(15 downto 0):=x"0000";

begin

anod\_0<="1110";

anod\_1<="1101";

anod\_2<="1011";

anod\_3<="0111";

cifra\_0<=intrare\_numarator(3 downto 0);

cifra\_1<=intrare\_numarator(7 downto 4);

cifra\_2<=intrare\_numarator(11 downto 8);

cifra\_3<=intrare\_numarator(15 downto 12);

counter: process(clk,counter\_display)

begin

if(rising\_edge(clk)) then counter\_display<=counter\_display+1;

end if;

end process;

selectie<=counter\_display(15 downto 14);

C1: mux4\_1 port map(cifra\_0,cifra\_1,cifra\_2,cifra\_3,selectie,iesire\_mux1);

C2: mux4\_1 port map(anod\_0,anod\_1,anod\_2,anod\_3,selectie,iesire\_anod);

C3: bcd\_7\_seg port map(iesire\_mux1,iesire\_catod);

anod<=iesire\_anod;

catod<=iesire\_catod;

end arth;

Proiectare ansamblu, codul comentat pentru ansamblu

Click\_counter

Componenta de ansamblu, numita **click\_counter**, include toate componentele comentate mai sus si imbinarea acestora pentru functionarea aplicației.

Aceasta componenta are ca intrari semnalul de **clk** de perioadă 10 ns, **reset** (care va fi folosit pentru resetare) , **buton\_st**, **buton\_dr**, **reverse** corespunzatoare celor trei butoane pentru incrementare, decrementare si respectiv inversarea rolurilor butoanelor. Are drept iesiri **anod, catod** și **is\_left** (ar fi fost reprezentat pe plăcuță de un led care se aprinde când butonul de reverse este activat) si doi vectori de 4, 7 elemente pentru cei 4 anozi, respectiv cei 7 catozi de pe fiecare anod(ar fi fost reprezentați pe ssd).

În această componentă conectăm toate componentele prezentate mai sus(numărător,display și uc) , prin port-maparea acestora

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity click\_counter is

Port ( clk : in STD\_LOGIC;

buton\_st : in STD\_LOGIC;

buton\_dr : in STD\_LOGIC;

reverse : in STD\_LOGIC;

reset : in STD\_LOGIC;

anod : out STD\_LOGIC\_VECTOR (3 downto 0);

catod : out STD\_LOGIC\_VECTOR (6 downto 0);

is\_left : out STD\_LOGIC);

end click\_counter;

architecture click\_counter of click\_counter is

component numarator is

port( increase: in STD\_LOGIC;

decrease: in STD\_LOGIC;

reset: in STD\_LOGIC;

clk: in STD\_LOGIC;

solutie: out STD\_LOGIC\_VECTOR (15 downto 0));

end component;

component display is

port(clk : in std\_logic;

intrare\_numarator: in STD\_LOGIC\_VECTOR(15 downto 0);

anod : out STD\_LOGIC\_VECTOR(3 downto 0);

catod : out STD\_LOGIC\_VECTOR(6 downto 0));

end component;

component uc is

port(clk:in std\_logic;

reset:in std\_logic;

reverse:in std\_logic;

buton\_st:in std\_logic;

buton\_dr:in std\_logic;

increase:out std\_logic;

decrease:out std\_logic;

is\_left:out std\_logic);

end component;

signal inc,dec: std\_logic;

signal sol: std\_logic\_vector(15 downto 0);

begin

unitate\_comanda: uc port map(clk,reset,reverse,buton\_st,buton\_dr,inc,dec,is\_left);

counter: numarator port map(inc,dec,reset,clk,sol);

ssd: display port map(clk,sol,anod,catod);

end click\_counter;

Lista de componente utilizate

* Numarare (numarator)
* Afisare (display)
* UC
* Filtru(nefolosit)

Specificatia notatiilor I/O si a semnalelor interne

Intrări:

buton\_st - buton pentru incrementare

clk - clock modul de simulare

buton\_dr - buton pentru decrementare

reset - buton de resetare

reverse - buton pentru inversarea funcțiilor butoanelor

Ieșiri:

is\_left - led pentru semnalizarea activării butonului reverse

anod - vector de 4 elemente pentru cei 4 anozi

catod - vector de 7 elemente pentru cele 7 segmente (catozi)

Semnale interne:

stare,nxstare – semnale folosite în UC pentru determinarea sensului de numărare

inc - semnal de enable, activ pe 1 logic pentru numararea crescatoare

dec - semnal de enable, activ pe 1 logic pentru numararea descrescătoare

cnt – semnal folosit în numărător pentru reținerea valorii numărate, acesta va fi atribuit ieșirii

sol – semnal folosit în click\_counter pentru reținerea valorii numărate, acesta va fi folosit ca ieșire din numărător și intrare în ssd

cifra\_0, cifra\_1, cifra\_2, cifra\_3, anod\_0, anod\_1, anod\_2, anod\_3 – semnale din display

Utilizare și rezultate

Resurse necesare

- programul **ISE Design Suite - Xilinx** sau **Vivado Design Suite - Xilinx**

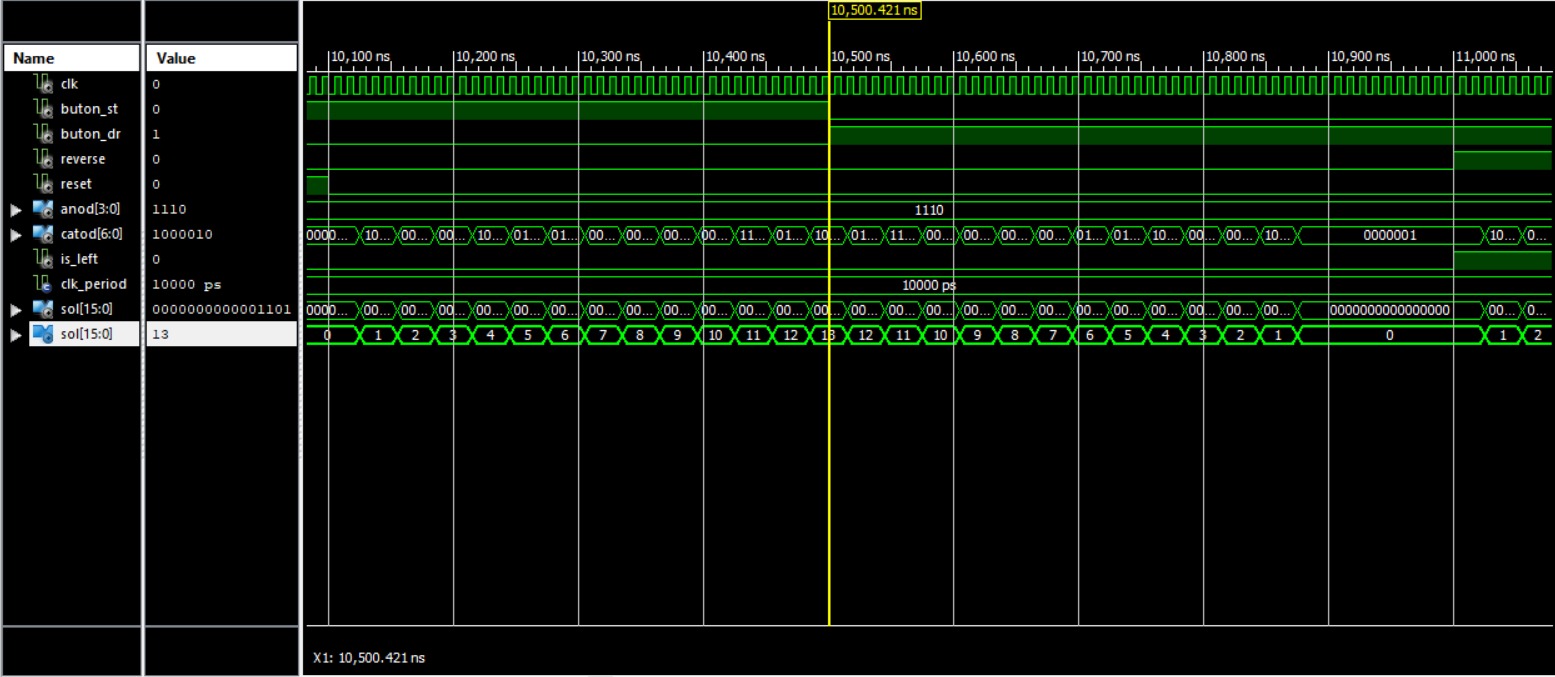
Descrierea utilizarii

În modulul de simulare se observă cum inițial sol are valoarea 0 (am pus-o atât în binar cât și în zecimal). Prin apăsarea butonului **buton\_st** începe numărarea crescătoare. Prin apăsarea butonului **buton\_dr** începe numărarea descrescătoare (până ajungem la 0 sau se apasă alt buton). În orice moment dacă se apasă butonul reset, numărarea începe dinnou de la 0. De asemenea, butonul reverse va inversa rolurile butonului stânga și butonului dreapta. Apăsarea acestui buton este semnalată prin aprinderea unui led.

Toate aceste instrucțiuni au fost făcute prestabilit în test bench

Rezultate obținute în modulul de simulare:

* am pus buton\_st pe 1, contorul crește, iar apoi buton\_dr pe 1 și contorul scade



* am verificat resetul în timpul numărării



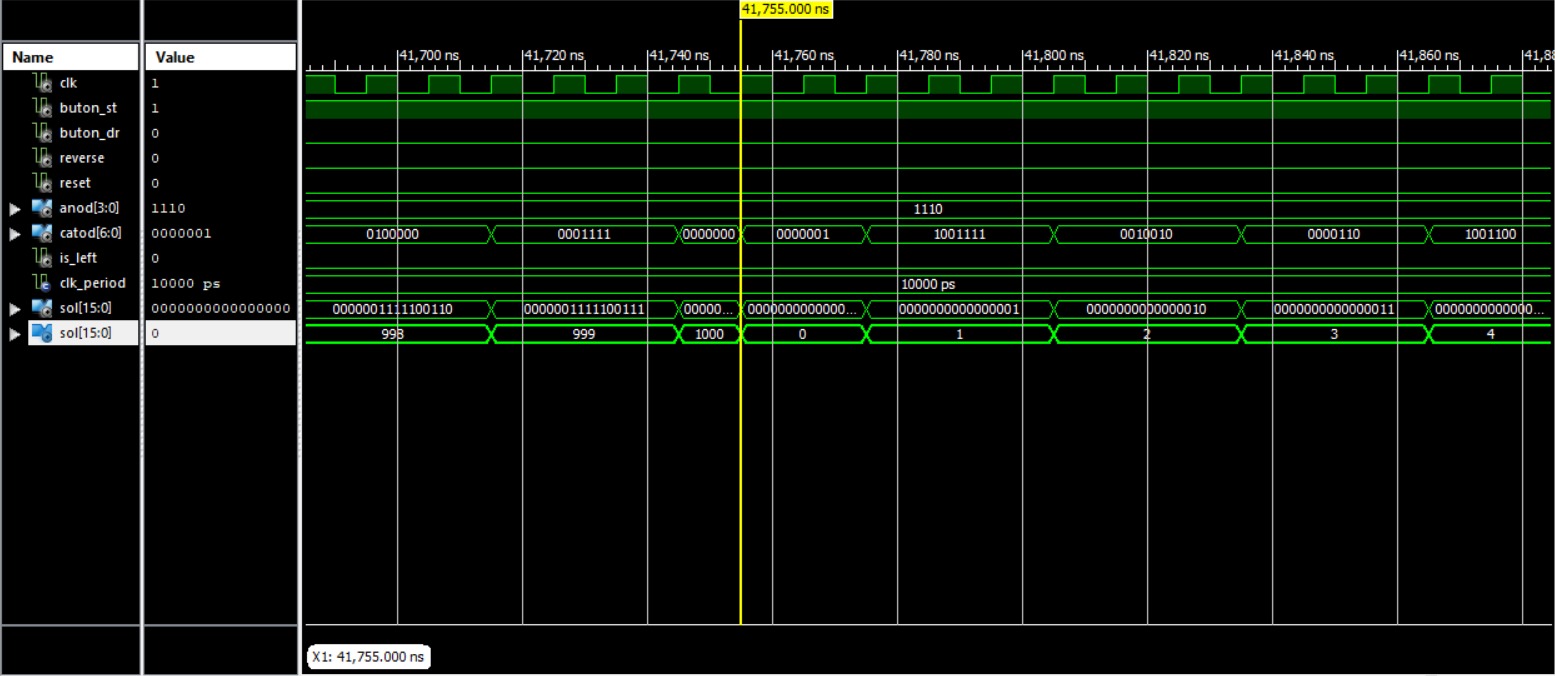
* la 11000 ns am activat reverse, se observă cum cu buton\_dr activat contorul crește



* la 12000 ns am dezactivat reverse-ul iar butoanele și-au reluat funcțiile inițiale



* contorul ajunge la 1000 și se resetează



Posibilitati de dezvoltare

Implementarea pe o plăcuță cu FPGA.

Utilizarea unui mouse real în schimbul butoanelor deja existente pe plăcuță.

Realizarea unui numărător mai complex, care ar putea număra din n în n.

Bibliografie

<https://users.utcluj.ro/~vcristian/AC.html>

<https://www.fpga4student.com/2017/09/vhdl-code-for-seven-segment-display.html>