**Documentatie Cronometru**

In implementarea cronometrului am folosit placa Nexys2 ca parte hardware, iar ca implementare software am folosit Xilinx ISE Design Suite si Adept. Cronometrulconstruit are functia de start/stop si reset.

De pe placa Nexys2 s-au folosit: cele 4 afisaje cu 7 segmente, un switch (start/stop)si un buton (reset). Totodata s-a folositsifrecventa de 50MHz data de oscilatorul cu cuart al placii Nexys2, care a fostdivizata in vedereaobtinerii a 2 factori importanti in functionarea corecta a cronometrului: o divizare a celor 50MHz astfel incat sa se obtina 10Hz (0.1s)si o divizare a aceleasi frecvente la valoarea 250Hz pentru a putea baleia intre cele 4 afisaje fara ca ochiul uman sa sesizeze aceasta baleiere.

Constructia programului a inceput prin declararea librariilor necesare a functionarii programului, intrarilor si a iesirilor portului.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity cronometru is

Port (clk: in STD\_LOGIC;

reset: in STD\_LOGIC;

stop: in STD\_LOGIC;

anode: out std\_logic\_vector(3 downto 0);

segments: out std\_logic\_vector (6 downto 0));

end cronometru;

Totodata a fost nevoie de definirea unor semnale care sa faca legatura intre diversele blocuri folosite in implementarea proiectului. Astfel s-au definit semnale care impreuna alcatuiesc blocuri ca: bloc de multiplexare (necesar pentru a face trecerea de la un afisaj cu 7 segmente la alta, sau mai precis spus comandarea anozilor celor 4 afisaje), blocul numaratorului, blocul codificatorului (necesar pentru a realize comanda segmentelor afisajelor, folosind la intrarea informatia data de numerator si furnizand la iesire informatia necesara pentru a forma pe afisajul cu 7 segmente cifra corespunzatoare) si nu in ultimul rand blocul divizorului de frecventa.

architecture cronometru of cronometru is

signal clk1: std\_logic:='0';

signal count: integer:=1;

signal div: std\_logic\_vector(22 downto 0);

signal alegereDisplay:std\_logic\_vector(1 downto 0);

signal digit1: INTEGER:= 0;

signal digit2: INTEGER:= 0;

signal digit3: INTEGER:= 0;

signal digit4: INTEGER:= 0;

signal digit5: INTEGER:= 0;

signal digit6: INTEGER:= 0;

type display\_ROM is array (0 to 9) of std\_logic\_vector (6 downto 0);

constant convert\_to\_segments: display\_ROM:=("1000000","1111001","0100100","0110000","0011001","0010010","0000010","1111000","0000000","0010000"); -- numerele 0,1,2,3...9

Ultima instructiune produce, intocmai, functia blocului condificatorului, transformand informatia primita la intrare, in secventa necesara afisarii de catre cele 4 afisaje a cifrelor de la 1 la 9.

PROCESS(clk1, reset)

BEGIN

IF (reset='1') THEN

digit1 <= 0;

digit2 <= 0;

digit3 <= 0;

digit4 <= 0;

digit5 <= 0;

digit6 <= 0;

ELSIF (rising\_edge(clk1) and stop= '0') THEN

digit1 <= digit1 + 1; --milisecunde

IF (digit1=9) THEN

digit1 <= 0;

digit2 <= digit2 + 1; --secunde

IF (digit2=9) THEN

digit2 <= 0;

digit3 <= digit3 + 1; -- secunde

IF (digit3=5) THEN --59 secunde, trecerea spre un minut

digit3 <= 0;

digit4 <= digit4 + 1; --minute

IF (digit4=9) THEN

digit4 <= 0;

digit5 <= digit5 + 1;

IF (digit5=9) THEN

digit5 <=0;

digit6 <= digit6 + 1;

if (digit6=3) then

digit1 <= 0;

digit2 <= 0;

digit3 <= 0;

digit4 <= 0;

digit5 <= 0;

digit6 <= 0;

END IF; END IF; END IF; END IF; END IF; END IF; END IF;

END PROCESS;

div<= div + 1 when rising\_edge(clk);

alegereDisplay<= div(16 downto 15);

Secventele de cod, de mai sus seteaza valorile celor 4 afisaje si sunt integrate intr-un proces. Daca butonul de reset este apasat toate afisajele vor afisa 0 (resetarea cronometrului). Daca butonul de stop nu este apasat si se intalneste frontal crescator al semnalului de tact atunci primul afisaj incepe incrementarea pana cand ajunge la valoarea 9, moment in care acesta trece pe 0, iar al 2-lea afisaj incepe incrementarea (tot pana la valoarea 9). In momentul in care si primul afisaj si al 2-lea afisaj ajung la valorea 9, incepe incrementarea celui de-al 3-lea afisaj, care va incrementa pana la valoarea 5 (pentru a forma 59.9 secunde). In momentul in care primele 3 afisaje arata valoarea 59.9 se va trece la numararea minutelor, astfel va incepe incrementarea si celui de-al 4-lea afisaj in functie de incrementarile primelor 3. Cand se va comuta switch-ul de start/stop, numaratorul se va opri si afisajele vor arata valoarea numarata pana in acel moment. La o a 2-a comanda a switch-ului start/stop numaratorul va continua sa numere de la valoarea de la care s-a dat stop.

Semnalul alegere Display indeplineste functia blocului de multiplexare comandand corespunzator anozii celor 4 afisaje astfel incat acestea sa afiseze in mod corespunzator (0.1s, apoi secundele, apoi minutele).

process(clk)

begin

if rising\_edge(clk) then

count <=count+1;

if(count = 2500000) then

clk1 <= not clk1;

count <=1;

end if;

end if;

if rising\_edge(clk) then

if alegereDisplay ="11" then

segments <= convert\_to\_segments(digit4); -- 0

anode<="0111";

elsifalegereDisplay ="10" then

segments <= convert\_to\_segments(digit3); -- 1

anode<="1011";

elsifalegereDisplay ="01" then

segments <= convert\_to\_segments(digit2); -- 2

anode<="1101";

else

segments <= convert\_to\_segments(digit1); -- 3

anode<="1110";

end if;

else null;

end if;

end process;

Al 2-lea process defineste blocul de divizare a frecventei (creeazacei 250Hz necesari pentru baleierea, insesizabila ochiului uman, intre afisaje). Totodata in acest process se prezinta si sintaxa corespunzatoare blocului de multiplexare (prin intermediul unor if-uri, in functie de informatia ajunsa la intrarea asa-zisului multiplexor, acesta ofera la iesire o secventa de biti care va comanda anozii celor 4 afisaje cu 7 segmente astfel incat afisajele sa se aprinda si sa afiseze intr-o ordine corespunzatoare).

Dupa rularea programului, s-a demonstrat ca acest cronometru functioneaza corect, fapt vizibil si in poza de mai jos:

