*Universitatea Tehnica Cluj-Napoca*

*Facultatea de Automatica si Calculatoare*

*Sectia Calculatoare*

*Anul 2017-2018*

*Semestrul II*

*PROIECTAREA SISTEMELOR NUMERICE*

**LIFT P+7**

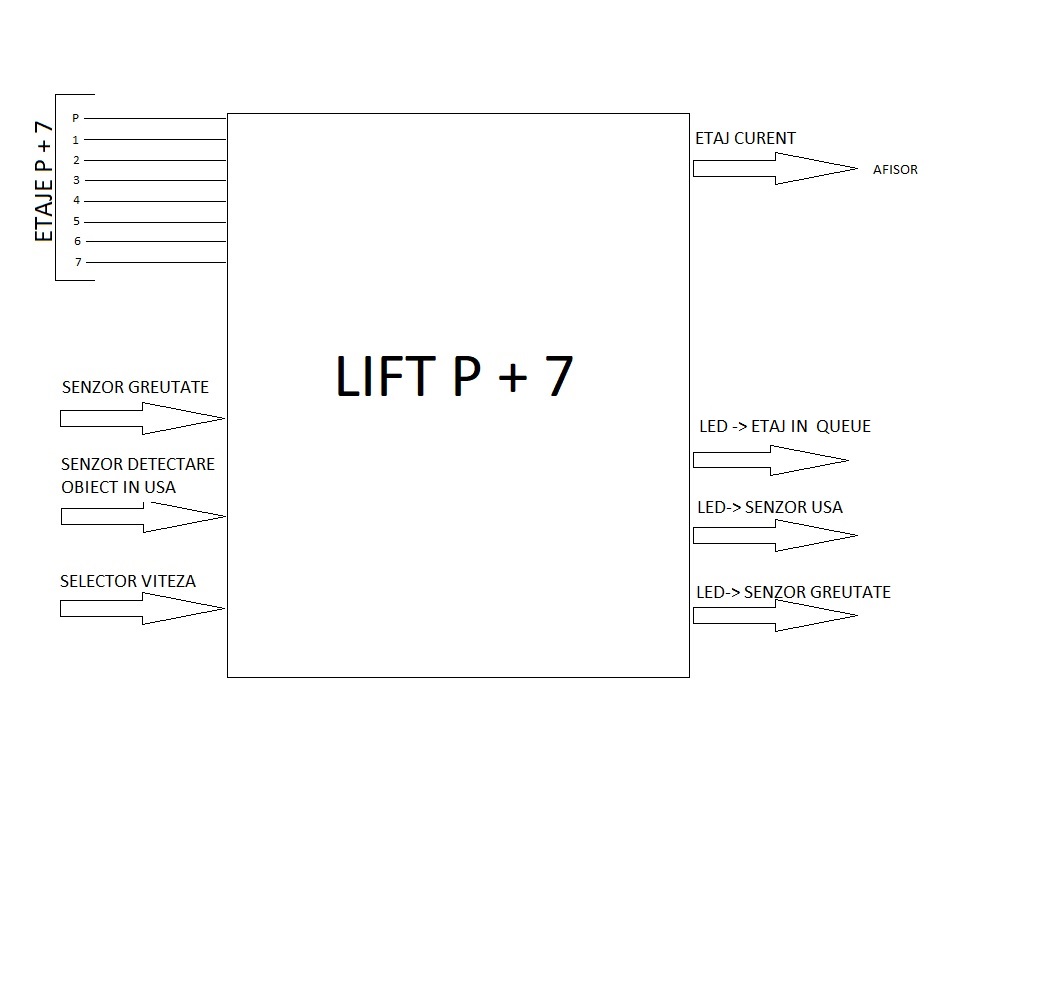
*Indrumator laborator: Cret Octavian.*

*Realizat de: Izota Andreea* ***&*** *Ulsamer Andreas-Mihail.*

***CUPRINS:***

1. Specificatia;
2. Schema bloc;
3. Intrari si iesiri**;**
4. Componente;
5. Instructiuni de utilizare;
6. Posibilitati de dezvoltare;
7. Justificarea solutiei alese.
8. **SPECIFICATIA**

Sa se proiecteze un automat care comanda un lift intr-un hotel P+7 etaje. Liftul trebuie sa raspunda solicitarilor persoanelor aflate in interior si cererilor exterioare care apar pe parcurs de la usile aflate la fiecare nivel. Ordinea de onorare a cererilor tine cont de sensul de mers (urcare sau coborare). Se onoreaza cererile in ordinea etajelor, indiferent de unde provin ele (lift sau exterior). Liftul are o intrare care sesizeaza depasirea greutatii maxime admise si nu porneste in acest caz. Plecarea nu are loc daca usile nu sunt inchise. Usile trebuie sa stea deschise un interval de timp programabil. Usile nu se inchid daca exista vreo persoana in usa. Viteza liftului va fi selectabila intre doua valori: 1 sau 3 secunde/etaj. Se considera ca in momentul initial liftul se gaseste la parter, cu usile deschise.

1. **SCHEMA BLOC**
2. **INTRARI SI IESIRI**

**INTRARI:**

* **clk** = Clock-ul placutei FPGA de 100MHz;
* **btn** = Vector de 5 elemente care reprezinta butoanele placutei FPGA;
* **sw** = Vector de 8 elemente care reprezinta etajele liftului, care se dau de pe switch-uri, fiecare etaj corespunzand unui switch;
* **sw\_aux** = Vector de 3 elemente care reprezinta switch-uri care au urmatoarele functionalitati: depasirea greutatii, usi deschise, selectarea vitezei.

**IESIRI:**

* **DP** = Dezactiveaza punctele de pe afisorul placutei FPGA;
* **LED** = Vector de 16 elemente care reprezinta ledurile placutei FPGA;
* **an** = Vector de 4 elemente care reprezinta anoadele afisorului folosit de pe placuta FPGA;
* **an\_aux** = Vector de 4 elemente care reprezinta anoadele afisorului nefolosit de pe placuta FPGA;
* **seg** = Vector de 7 elemente care reprezinta catoadele afisorului.

1. **COMPONENTE**

Liftul are urmatoarele componente: divizor de frecventa cu parametri generici (deoarece viteza este selectabila), debouncer, afisorul si liftul propriu-zis.

* **DIVIZORUL DE FRECVENTA**

Deoarece proiectul a fost conceput pentru o placuta **Nexys4**, divizorul de frecventa are rolul de a transforma cei *100MHz* de la placuta intr-o secunda, respectiv in 3 secunde.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

entity DivFrec is Generic (n : **integer**);

Port ( clk : in **STD\_LOGIC**;

clkout : out **STD\_LOGIC**);

end DivFrec;

architecture arh\_DivFrec of DivFrec is

signal temp : **integer** range 0 to n\*50\_000\_000 := 0;

begin

process(clk)

begin

if rising\_edge(clk) then

temp <= temp + 1;

end if;

if temp = 0 then

clkout <= '1';

else

clkout <= '0';

end if;

end process;

end arh\_DivFrec;

* **DEBOUNCER**

Pentru ca la apasarea butoanelor placutei sa nu fie introduse mai multe comenzi, se foloseste un debouncer.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity mpg is

Port (clk : in **STD\_LOGIC**;

btn : in **STD\_LOGIC\_VECTOR**(4 DOWNTO 0);

bt : out **STD\_LOGIC\_VECTOR**(4 DOWNTO 0));

end mpg;

architecture Behavioral of mpg is

signal count : **std\_logic\_vector** (17 downto 0);

signal d1, d2, d3 : **std\_logic\_vector**(4 downto 0);

begin

process (clk)

begin

if rising\_edge(clk) then

count <= count + 1;

if count = 0 then

d1 <= btn;

end if;

d2 <= d1;

d3 <= d2;

end if;

end process;

bt <= d2 and (not d3);

end Behavioral;

* **AFISOR**

Folosit pentru afisarea etajelor pe afisorul 7-segmente.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Afisor is

port(

digit0: in **std\_logic\_vector**(3 downto 0);

digit1: in **std\_logic\_vector**(3 downto 0);

digit2: in **std\_logic\_vector**(3 downto 0);

digit3: in **std\_logic\_vector**(3 downto 0);

CLK : in **std\_logic**;

seg: out **std\_logic\_vector**(0 to 6);

ano : out **std\_logic\_vector**(3 downto 0)

);

end Afisor;

architecture arh\_Afisor of Afisor is

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

signal aux : **std\_logic\_vector**(3 downto 0);

begin

process(CLK)

begin

if rising\_edge( CLK ) then

cnt <= cnt + 1;

end if;

end process;

process(cnt(15), cnt(14))

begin

case cnt(15 downto 14) is

when "00" => ano <= "1111";

when "01" => ano <= "1111";

when "10" => ano <= "1111";

when "11" => ano <= "0111";

when others => ano <= "0000";

end case;

end process;

process(cnt(15), cnt(14), digit0, digit1, digit2, digit3)

begin

case cnt( 15 downto 14 ) is

when "00" => aux <= digit0;

when "01" => aux <= digit1;

when "10" => aux <= digit2;

when "11" => aux <= digit3;

when others => aux <= "1111";

end case;

end process;

process (aux)

begin

case aux is

when "0000" => seg <="0001100"; --P

when "0001" => seg <="1111001"; --1

when "0010" => seg <="0100100"; --2

when "0011" => seg <="0110000"; --3

when "0100" => seg <="0011001"; --4

when "0101" => seg <="0010010"; --5

when "0110" => seg <="0000010"; --6

when "0111" => seg <="1111000"; --7

when others => seg <="1111111"; --

end case;

end process;

end arh\_Afisor;

* **LIFTUL PROPRIU-ZIS**

Descrierea comportamentului liftului.

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Main is

Port (

clk : in **STD\_LOGIC**;

btn : in **STD\_LOGIC\_VECTOR** (4 downto 0);

sw : in **STD\_LOGIC\_VECTOR** (7 downto 0);

sw\_aux : in **STD\_LOGIC\_VECTOR** (15 downto 13);

DP : out **STD\_LOGIC**;

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

an : out **STD\_LOGIC\_VECTOR** (0 to 3);

an\_aux : out **STD\_LOGIC\_VECTOR** (0 to 3);

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

end Main;

architecture Behavioral of Main is

component Mpg is

Port (clk : in **STD\_LOGIC**;

btn : in **STD\_LOGIC\_VECTOR** (4 downto 0);

bt : out **STD\_LOGIC\_VECTOR** (4 downto 0);

end component;

component DivFrec is Generic (n : **integer**);

Port ( clk : in **STD\_LOGIC**;

clkout : out **STD\_LOGIC**);

end component;

component Afisor is

port(

digit0: in **STD\_LOGIC\_VECTOR** (3 downto 0);

digit1: in **STD\_LOGIC\_VECTOR** (3 downto 0);

digit2: in **STD\_LOGIC\_VECTOR** (3 downto 0);

digit3: in **STD\_LOGIC\_VECTOR** (3 downto 0);

CLK : in **STD\_LOGIC**;

seg: out **STD\_LOGIC\_VECTOR** (0 to 6);

ano : out **STD\_LOGIC\_VECTOR** (3 downto 0)

);

end component;

signal bt : **STD\_LOGIC\_VECTOR** (4 downto 0);

signal print : **STD\_LOGIC\_VECTOR** (15 downto 0);

signal clk1, clk3, greutate, piu, func, modmers, led1, led3 : **STD\_LOGIC**;

signal NrStationare : **integer** range 0 to 10 := 0;

type coada is array (8 downto 0) of **STD\_LOGIC\_VECTOR** (2 downto 0);

signal Q : coada;

signal Qcnt : **STD\_LOGIC\_VECTOR** (3 downto 0) := "1111";

signal EtajCurent : **STD\_LOGIC\_VECTOR** (2 downto 0) := "000";

signal Qmark : **STD\_LOGIC\_VECTOR** (7 downto 0) := "00000000";

begin

greutate <= sw\_aux(15);

piu <= sw\_aux(14);

modmers <= sw\_aux(13);

led(7 downto 0) <= Qmark;

led(10 downto 8) <= "000";

led(15) <= greutate;

led(14) <= piu;

led(13 downto 11) <= "000";

func <= (not greutate) and (not piu);

print(3 downto 0) <= "0" & EtajCurent;

DP <= '1';

Afisor1 : Afisor port map(print(15 downto 12), print(7 downto 4), print(11 downto 8), print(3 downto 0), clk, seg, an);

an\_aux <= "1111";

Mpg1 : Mpg port map(clk, btn, bt);

DivFrec1 : DivFrec generic map(n => 1) port map(clk, clk1);

DivFrec2 : DivFrec generic map(n => 3) port map(clk, clk3);

process (clk)

begin

if rising\_edge(clk) then

if bt(0) = '1' then *--Coada comenzi*

if (sw(7 downto 0) and (sw(7 downto 0) - 1)) = 0 and (sw(7 downto 0) /= 0) then *--Doar un switch ridicat* *=>* x & (x-1) == 0

if sw(7) = '1' and Qmark(7) /= '1' then

Q(0) <= "111";

Qmark(7) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(6) = '1' and Qmark(6) /= '1' then

Q(0) <= "110";

Qmark(6) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(5) = '1' and Qmark(5) /= '1' then

Q(0) <= "101";

Qmark(5) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(4) = '1' and Qmark(4) /= '1' then

Q(0) <= "100";

Qmark(4) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(3) = '1' and Qmark(3) /= '1' then

Q(0) <= "011";

Qmark(3) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(2) = '1' and Qmark(2) /= '1' then

Q(0) <= "010";

Qmark(2) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(1) = '1' and Qmark(1) /= '1' then

Q(0) <= "001";

Qmark(1) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

elsif sw(0) = '1' and Qmark(0) /= '1' then

Q(0) <= "000";

Qmark(0) <= '1';

Qcnt <= Qcnt + 1;

Q( 7 downto 1 ) <= Q( 6 downto 0 );

end if;

end if;

end if;

if modmers = '0' then

if( clk1 = '1' and func = '1') then

if( NrStationare = 5 ) then

if( Qcnt < "1111" ) then

if( Q(conv\_integer(Qcnt)) > EtajCurent ) then

EtajCurent <= EtajCurent + 1;

elsif( Q(conv\_integer(Qcnt)) < EtajCurent ) then

EtajCurent <= EtajCurent - 1;

else

Qmark(conv\_integer(Q(conv\_integer(Qcnt)) ) ) <= '0';

Qcnt <= Qcnt - 1;

NrStationare <= 0;

end if;

end if;

else

NrStationare <= NrStationare + 1;

end if;

end if;

else

if( clk3 = '1' and func = '1') then

if( NrStationare = 5 ) then

if( Qcnt < "1111" ) then

if( Q(conv\_integer(Qcnt)) > EtajCurent ) then

EtajCurent <= EtajCurent + 1;

elsif( Q(conv\_integer(Qcnt)) < EtajCurent ) then

EtajCurent <= EtajCurent - 1;

else

Qmark(conv\_integer( Q(conv\_integer(Qcnt)) ) ) <= '0';

Qcnt <= Qcnt - 1;

NrStationare <= 0;

end if;

end if;

end if;

elsif(NrStationare < 5 and clk1 = '1' and func = '1') then

NrStationare <= NrStationare + 1;

end if;

end if;

end if;

end process;

end Behavioral;

1. **INSTRUCTIUNI DE UTILIZARE**

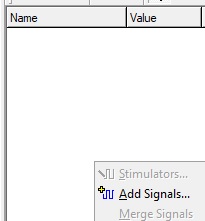
* **Active HDL**

Pentru a rula proiectul in **Active HDL** se vor efectua urmatorii pasi:

* Se deschide **Active HDL**;
* Se compileaza (apasati pe butonul „Compile”);



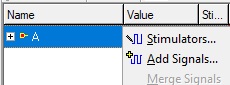
* Se simuleaza (apasati pe butonul „New Waveform”);
* In fereastra nou deschisa, se face click dreapta si se apasa pe butonul „Add signals”;



* Se adauga toate intrarile, iesirile si semnalele

apasand pe „Select All Signals”;

* Se dau valori intrarilor apasand click dreapta si apoi selectand butonul „Stimulators”. (Selectati „Hotkey” pentru switch-uri/butoane si „Clock” pentru clock-urile proiectului);



* Se apasa butonul „Run for”;
* Se vor selecta valorile pentru rulare, apoi se vor urmari rezultatele.
* **FPGA & XILINX**

Pentru a implementa proiectul pe **Nexys4** se vor efectua urmatorii pasi:

* Se ruleaza **XILINX ISE FOUNDATION**;
* Se creeaza un nou Workspace;
* Click pe „User Constraints”
* Select „Edit Constraints” (Locul unde input-urile primesc butoane (.ucf));
* Click pe „Generate Programming File”;
* In fereastra nou deschisa, click „Initialize Chain”;
* Click pe „Browse” si cauta fisierul cu extensia .bit;
* Click pe „Program”;
* Testati proiectul pe placa.

1. **POSIBILITATI DE DEZVOLTARE**

Liftul poate fi imbunatat adaugand 2 butoane de anularea comenzii, unul situat in exterior si unul in interior, astfel se economiseste energie si timp deoarece liftul nu mai urca/coboara la etajul gresit. De asemenea, o alta imbunatatire ar fi afisajul unei cozi cu etajele la care este chemat, astfel utilizatorul o sa stie cate opriri face liftul pana la el.

Pentru siguranta utilizatorilor liftului se poate adauga un buton in interior care, dupa apasarea lui, va initializa un apel de urgenta la 112. Astfel in cazului unui blocaj, utilizatorul poate cere ajutor imediat. Din motive de siguranta, butonul va fi amplasat intr-un colt, departe de celelalte butoane si va trebui apasat de 5 ori consecutiv ca sa functioneze.

1. **JUSTIFICAREA SOLUTIEI ALESE**

Pentru eficienta si simplitate am ales dezvoltarea acestui proiect in mod comportamental. Astfel, dupa cateva curse cu liftul, am putut pune pe foaie comportamentul lui mult mai usor si mai practic. De asemenea am limitat numarul de componente alese, avand Afisor BCD, Debounce si Divizor de frecventa, pe care le utilizam pentru descrierea propiu-zisa a liftului.

Liftul este un automat foarte util, in cladiri mari (pentru parcurgerea etajelor mult mai rapid), dar si pentru persoane cu dizabilitati care, de altfel, nu ar putea folosi scarile.

