



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA: Automatica si Calculatoare

SPECIALIZAREA: Calculatoare si tehnologia informatiei

DISCIPLINA: Proiectarea sistemelor numerice

PROIECT: LIFT P+12

Profesor coordonator: Opincariu Laurentiu

Studenti: Berengea Cristian

Ghiura Darius

Grupa: 30217

CUPRINS

- 1. Specificatie proiect**
- 2. Exemplu de functionare**
- 3. Schema bloc**
 - 3.1. Cutie neagra**
 - 3.2. Lista intrari si iesiri**
 - 3.3. Schema in detaliu**
 - 3.4. Lista semnale intermediare**
- 4. Proiectare si implementare**
 - 4.1. Lista componente**
 - 4.2. Codul componentelor comentat**
 - 4.3. Lift (top-level)**
- 5. Justificarea solutiei alese**
- 6. Instructiuni de utilizare**
- 7. Posibilitati de dezvoltare**

1. Specificatie proiect

Sa se proiecteze un automat care comanda un lift intr-un hotel cu P+12 etaje. Liftul trebuie sa raspunda solicitarilor persoanelor aflate in interior si cererilor exterioare (sus, jos) care apare 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.

2. Exemplu de functionare

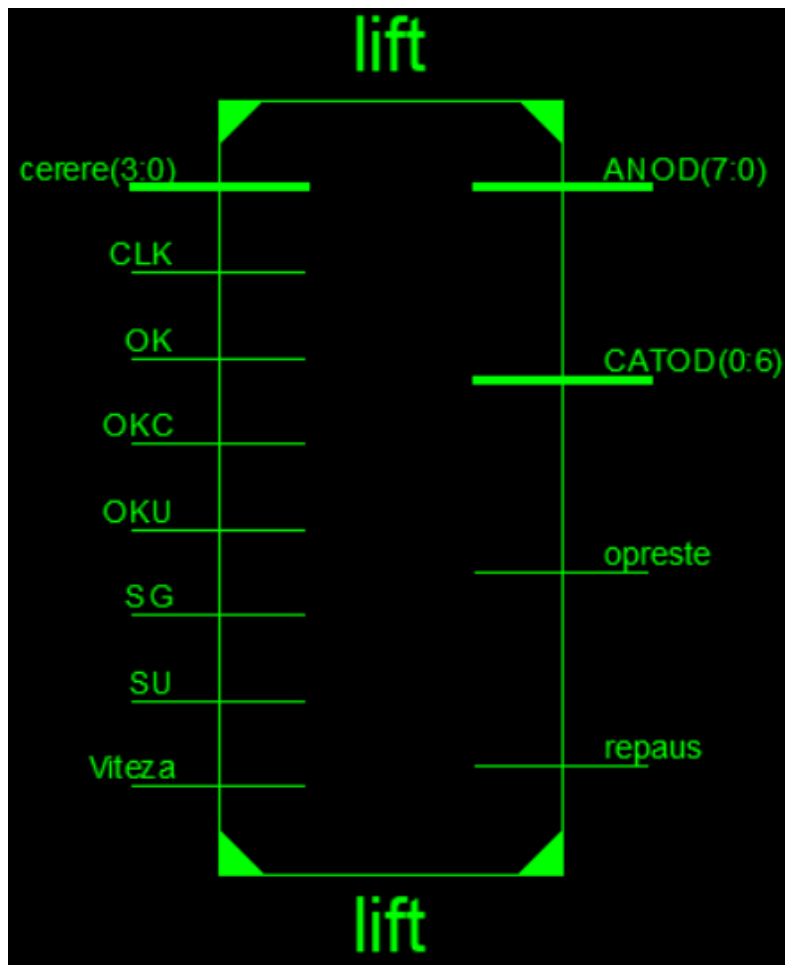
Liftul onoreaza cererile in functie de sensul de mers. In interiorul liftului exista 13 butoane pentru selectare etajului. De asemenea, la fiecare etaj sunt 2 butoane : unul pentru urcare ▲ si unul pentru coborare ▼ .

Persoanele aflate in interiorul liftului vor apasa pe butonul corespunzător etajului la care doresc ajunga. Persoanele aflate in exteriorul liftului vor apasa pe butonul corespunzator conform sensului dorit.

Sa consideram urmatorul scenariu: o persoana aflata la etajul 7 cheama liftul pentru a cobori la etajul 3, intre timp o alta persoana aflata la etajul 5 cheama liftul pentru a cobori la etajul 2. Liftul pleaca de la parter urca pana la etajul 7 fara sa opreasca la etajul 5. Dupa ce a ajuns la etajul 7, liftul isi schimba sensul de deplasare si doar atunci opreste la etajul 5, apoi opreste in 3 si in 2, iar dupa ce nu mai exista comenzi, liftul este programat sa se intoarca la parter si sa ramana cu usile deschise.



3. Schema bloc

3.1 Cutie neagra



3.2 Lista intrari si iesiri

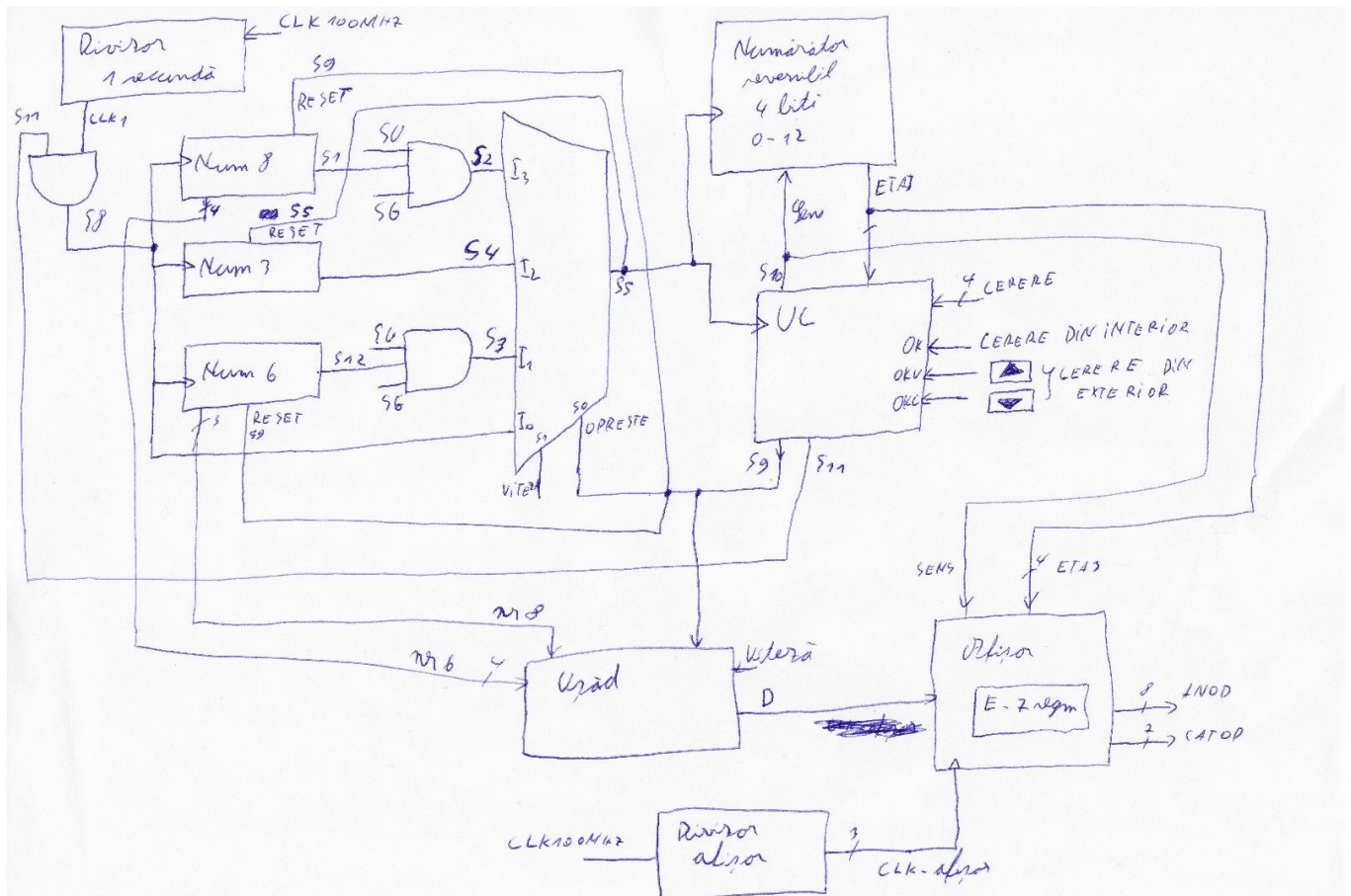
Intrari:

- cerere (3:0) - primeste valorile de la 4 switch-uri pentru introducerea unei cereri
- CLK – Semnalul de tact al placii (100 MHz)
- OK – Buton pentru inregistrarea cererii din interiorul liftului
- OKC – Buton pentru inregistrarea cererii din exteriorul liftului pentru coborare

- OKU – Buton pentru inregistrarea cererii din exteriorul liftului pentru urcare

- SG – Primeste valoarea de la un switch care reprezinta senzorul de greutate
- SU – Primeste valoarea de la un switch care reprezinta senzorul ce detecteaza daca e cineva in usa
- Viteza – Primeste valoarea de la un switch care reprezinta viteza liftului de trecere de la un etaj la altul (1 sau 3 secunde / etaj)

Iesiri:

- ANOD (7:0) - Activeaza anozii afisorului de pe placa
- CATOD (0:6) - Activeaza catozii afisorului de pe placa (7 segment)
- opreste - Semnal care activeaza un led de pe placa daca liftul opreste la etajul curent
- repaus – Semnal care activeaza un led de pe placa daca exista comenzi

3.3 Schema in detaliu



3.4 Lista semnale intermediare

- S1 - Terminal Count de la num8, intra intr-o poarte SI cu 3 intrari
- S2 - Iesire de la poarta SI cu 3 intrari, intra in MUX pe I3
- S3 - Iesire de la poarta SI cu 3 intrari, intra in MUX pe I1
- S4 - Terminal Count de la num3, intra in MUX pe I2
- S5 – Iesire de la MUX, actioneaza ca RESET pentru num3, CLK pentru Numaratorul reversibil si Unitatea de Comanda
- S8 - Transmite semnalul de tact divizat daca exista comenzi

- S9 - Iesire din UC, actioneaza ca RESET pentru num6, num8, Selectia S0 pentru MUX, este activ cand liftul opreste la etajul respectiv
- S10 - Iesire din UC, reprezinta sensul de deplasare al liftului (0 – urcare, 1 – coborare) , actioneaza ca Mod de numarare pentru numaratorul reversibil
- S11 - Iesire din UC, este 0 cand nu sunt comenzi
- S12 - Terminal Count de la num6, intra intr-o poarte SI cu 3 intrari
- Et - Iesire din numaratorul reversibil, reprezinta etajul curent, intrare pentru UC si pentru Afisor
- CLK1 - Semnalul de clock divizat
- CLK_afisor - Semnal de tact pentru afisor
- nr6 - Stare a lui num6
- nr8 - Stare a lui num8
- D – Iesire de la usad, este activ cand usa este deschisa, intra in afisor

4. Proiectare si implementare

4.1 Lista componente

1. Divizoare de frecventa
2. Poarta si cu 2 intrari
3. Poarta si cu 3 intrari
4. Numaratoare
5. Multiplexor
6. Unitate de Comanda
7. Decodificator
8. Afisor

4.2 Codul componentelor comentat

Am folosit libraria IEEE si pachetele 1164 si unsigned inainte de fiecare entitate.

```

library IEEE;

use IEEE.std_logic_unsigned.all;

use IEEE.std_logic_1164.all;

```

1. Divizor de frecventa (1 secunda)

```

entity div1 is
    port ( CLK : in std_logic;
          D : out std_logic);
end div1;

architecture Adiv1 of div1 is
    signal S : std_logic_vector (26 downto 0):= "000000000000000000000000";
begin
    process (CLK)
    begin
        if (CLK'EVENT) and (CLK='1') then S<= S+1;
        end if;
        D <= S(26);
    end process;
end Adiv1;

```

Divizeaza frecventa semnalului de tact al placii de la 100 MHz la 1 Hz (1 secunda) .

2. Poarta si cu 2 intrari

```

entity si2 is
    port(A,B: in std_logic;
          Y: out std_logic);

```


end si2;

architecture Asi2 **of** si2 **is**

begin

Y <= A and B;

end Asi2;

3. Numarator % 8

entity num8 **is**

port (CLK,RESET: **in** std_logic;

Q: **out** std_logic;

nr:**out** std_logic_vector(3 **downto** 0));

end num8;

architecture Anum8 **of** num8 **is**

begin

process (CLK,RESET)

variable Y: STD_LOGIC_vector(3 **downto** 0) := "0000";

begin

if RESET= '0' **then** Y:="0000";

end if;

if (CLK'EVENT) and (CLK = '1') **then** Y := Y + "0001";

if Y="1000" **then**

Q<='1';

Y:="0000";

else Q<='0';

end if;

end if;

nr<=Y;

end process;

end Anum8;

Folosim acest numarator % 8 cand liftul are viteza de 3 secunde / etaj si trebuie sa opreasca la etajul respectiv. Reset-ul este dat de semnalul opreste care iese din UC si este activ pe 0 deoarece atunci cand liftul a ajuns la etajul unde trebuie sa opreasca, semnalul opreste devine 1 astfel numaratorul numara 8 secunde (5 secunde pentru usi deschide si 3 pentru trecerea la etajul urmator).

4. Numarator % 6

entity num6 **is**

port (CLK,RESET: **in** std_logic;

Q: **out** std_logic;

nr:**out** std_logic_vector(2 **downto** 0));

end num6;

architecture Anum6 **of** num6 **is**

begin

process (CLK,RESET)

variable Y: STD_LOGIC_vector(2 **downto** 0) := "000";

begin

if RESET= '0' **then** Y:="000"; **end if**;

if (CLK'EVENT) and (CLK = '1') **then** Y := Y + "001";

if Y="110" **then**

Q<='1';

Y:="000";

else Q<='0';

end if;

end if;

nr<=Y;

end process;

end Anum6;

Folosim acest numarator % 6 cand liftul are viteza de 1 secunda / etaj si trebuie sa opreasca la etajul respectiv. Reset-ul este dat de semnalul opreste care iese din UC si este activ pe 0 deoarece atunci cand liftul a ajuns la etajul unde trebuie sa opreasca, semnalul opreste devine 1 astfel numaratorul numara 6 secunde (5 secunde pentru usi deschide si 1 pentru trecerea la etajul urmator).

5. Poarta si cu 3 intrari

```
entity si3 is
    port(A,B,C: in std_logic;
        Y: out std_logic);
end si3;
```

```
architecture Asi3 of si3 is
    signal X: std_logic;
begin
    X<= A and B;
    Y <= X and C;
    end Asi3;
```

6. Numarator % 3

```
entity num3 is
    port (CLK,RESET: in std_logic;
        Q: out std_logic);
end num3;

architecture Anum3 of num3 is
begin
    process (CLK,RESET)
        variable Y: STD_LOGIC_vector(1 downto 0) := "00";
    begin
        if RESET= '1' then Y:="00";
```

```

    end if;

    if (CLK'EVENT) and (CLK = '1') then Y := Y + "01";

    if Y="11" then

        Q<='1';

        Y:="00";

    else Q<='0';

    end if;

end if;

end process;

end Anum3;

```

Folosim acest numarator % 3 cand liftul are viteza de 3 secunde / etaj si nu trebuie sa opreasca la etajul respectiv. Se reseteaza de fiecare data cand numaratorul reversibil isi schimba stare (liftul trece la etajul urmator) .

7. Multiplexor 4 la 1

```

entity mux41 is

    port (I0,I1,I2,I3,S0,S1:in std_logic;

        Q: out std_logic);

end mux41;

architecture Amux41 of mux41 is

begin

    process(I0,I1,I2,I3,S0,S1)

    begin

        if (S0 = '0') and (S1 = '0') then Q<=I0;

        elsif (S0 = '1') and (S1 = '0') then Q<=I1;

        elsif (S0 = '0') and (S1 = '1') then Q<=I2;

        else Q<=I3;

        end if;

    end process;

end Amux41;

```

Este folosit pentru a selecta semnalul de tact al numaratorului reversibil, se disting cele 4 cazuri:

- Viteza 1 sec si nu opreste, caz in care folosim clock-ul divizat
- Viteza 1 sec si opreste, caz in care folosim numaratorul % 6
- Viteza 3 sec si nu opreste, caz in care folosim numaratorul % 3
- Viteza 3 sec si opreste, caz in care folosim numaratorul % 8

8. Unitatea de Comanda

entity UC is

```
port (  
    OK,OKU,OKC,CLK:in std_logic;  
    NR: in std_logic_vector (3 downto 0);  
    COM: in std_logic_vector (3 downto 0);  
    sens,opreste,repaus: out std_logic);
```

end UC;

architecture AUC of UC is

```
signal a_sens: std_logic := '0' ;
```

begin

```
process (CLK,OK,OKU,OKC)
```

```
variable v: natural;
```

```
variable v1: natural;
```

```
variable e_h: integer :=-1;
```

```
variable e_l: integer :=13;
```

```
variable urcare: std_logic_vector (0 to 12):= "00000000000000";
```

```
variable coborare: std_logic_vector (0 to 12):= "00000000000000";
```

```
begin
```

```
    v:= CONV_INTEGER(NR);
```

```
    v1:= CONV_INTEGER(COM);
```

```
if (OKC = '1') or (OK = '1') or (OKU = '1') then
```

```
if (OKC = '1') then -- adaugare comanda
```

```

coborare(v1):='1';
repaus<='1';
if (e_l=13) and (e_h=-1) and (a_sens = '1') and (v1>v) then a_sens<='0'; end if;
end if;
if (OK = '1') then -- adaugare comanda

    if (v1<v) then coborare(v1):='1';
    else urcare(v1):='1'; end if;

repaus<='1';

if (e_l=13) and (e_h=-1) and (a_sens = '1') and (v1>v) then a_sens<='0'; end if;
end if;
if (OKU = '1') then -- adaugare comanda
    urcare(v1):='1';
    repaus<='1';

if (e_l=13) and (e_h=-1) and (a_sens = '1') and (v1>v) then a_sens<='0'; end if;
end if;
else
if urcare(12)='1' or coborare(12)='1' then e_h:=12;
    elsif urcare(11)='1' or coborare(11)='1' then e_h:=11;
    elsif urcare(10)='1' or coborare(10)='1' then e_h:=10;
    elsif urcare(9)='1' or coborare(9)='1' then e_h:=9;
    elsif urcare(8)='1' or coborare(8)='1' then e_h:=8;
    elsif urcare(7)='1' or coborare(7)='1' then e_h:=7;
    elsif urcare(6)='1' or coborare(6)='1' then e_h:=6;
    elsif urcare(5)='1' or coborare(5)='1' then e_h:=5;
    elsif urcare(4)='1' or coborare(4)='1' then e_h:=4;
    elsif urcare(3)='1' or coborare(3)='1' then e_h:=3;
    elsif urcare(2)='1' or coborare(2)='1' then e_h:=2;
    elsif urcare(1)='1' or coborare(1)='1' then e_h:=1;
    elsif urcare(0)='1' or coborare(0)='1' then e_h:=0;
    else e_h:=-1;

```

end if;

```
if urcare(0)='1' or coborare(0)='1' then e_l:=0;  
elsif urcare(1)='1' or coborare(1)='1' then e_l:=1;  
elsif urcare(2)='1' or coborare(2)='1' then e_l:=2;  
elsif urcare(3)='1' or coborare(3)='1' then e_l:=3;  
elsif urcare(4)='1' or coborare(4)='1' then e_l:=4;  
elsif urcare(5)='1' or coborare(5)='1' then e_l:=5;  
elsif urcare(6)='1' or coborare(6)='1' then e_l:=6;  
elsif urcare(7)='1' or coborare(7)='1' then e_l:=7;  
elsif urcare(8)='1' or coborare(8)='1' then e_l:=8;  
elsif urcare(9)='1' or coborare(9)='1' then e_l:=9;  
elsif urcare(10)='1' or coborare(10)='1' then e_l:=10;  
elsif urcare(11)='1' or coborare(11)='1' then e_l:=11;  
elsif urcare(12)='1' or coborare(12)='1' then e_l:=12;  
else e_l:=13;  
end if;
```

if (CLK = '0') and (CLK'EVENT) **then**

if (a_sens = '0') **then**

```
if urcare(v) = '1' or v=e_h then opreste <= '1';  
  else opreste <= '0'; end if;  
  urcare(v):='0';
```

if (v >= e_h) **then** a_sens <= '1'; coborare(v):='0'; **end if;**

if (e_l=13) and (e_h=-1) **then** a_sens <= '1'; coborare(v):='0'; **end if;**

else

```
if coborare(v) = '1' or v=e_l then opreste <= '1';  
else opreste <= '0'; end if;  
  coborare(v):='0';
```

```

    if (v = e_l) and (e_l < e_h) then a_sens <= '0'; urcare(v):='0';
    elsif (e_h = e_l) and (v < e_l) then a_sens <= '0'; urcare(v):='0';
    end if;
end if;

if(v=0) and (e_l<13) and (a_sens = '1') then a_sens<='0'; end if;

end if;

-- if (e_l=0) and (e_h=0) and (v = 0) then urcare(v):='0'; coborare(v):='0'; end if;
if (e_l=13) and (e_h=-1) and (CLK = '0') and (v = 0) -- stabilire stare de repaus, parter
    then repaus<='0'; a_sens <= '0'; opreste <= '1';
    else repaus<='1';
    end if;

end if;

end process;

sens <= a_sens;

end AUC;

```

Unitatea de Comanda inregistreaza si proceseaza comenzile, stabileste sensul de deplasare si daca liftul opreste sau nu. Comenzile sunt memorate in 2 registri (**std_logic_vector**), unul pentru urcare si unul pentru coborare. Cand o cerere este inregistrata in registrul respectiv, la indexul comenzii punem valoarea 1, iar atunci cand liftul ajunge la etajul respectiv, comanda este stearsa punand in registrul, la pozitia etajului valoarea 0. Comenzile se proceseaza in functie de sensul de mers, nu in functie de aparitie.

Comenzile date din exterior vor intra in registrul potrivit, in functie de butonul apasat, iar din interior etajul curent se compara cu cererea pentru a stabili registrul in care sa fie memorata cererea. Daca cererea introdusa este mai mare decat etajul curent, ea va fi memorata in registrul pentru urcare, altfel va fi memorata in registrul pentru coborare.

Liftul se va deplasa pe un sens de mers pana ajunge in comanda din extremitate, unde isi va schimba sensul. Extremitatea este determinata verificand ambii registrii. Extremitatea de sus reprezinta cea mai de sus comanda, indiferent in ce

registru se afla. Extremitatea de jos reprezinta cea mai de jos comanda, indiferent in ce registru se afla.

Daca liftul urca sunt onorate comenzile din registrul de urcare, altfel din cel de coborare. Liftul nu opreste daca comanda este facuta pe sensul opus celui de deplasare, doar dupa ce ajunge in extremitate si isi schimba sensul. De exemplu: daca liftul se afla la parter si primeste o comanda din interior la etajul 11, o comanda din exterior pentru coborare la etajele 7 si apoi 8. Liftul va urca pana la etajul 11 fara sa opreasca in 7 si in 8, apoi isi va schimba sensul si va opri in 8 apoi in 7 si apoi se va intoarce la parter, in stare de repaus.

Liftul se va intoarce la parter daca nu mai exista comenzi.

9. Numarator reversibil pe 4 biti

entity rnum4 **is**

port(CLKK,M: **in** std_logic;

 Q: **out** std_logic_vector (3 **downto** 0));

end rnum4;

architecture rnr4 **of** rnum4 **is**

signal Y: STD_LOGIC_vector(3 **downto** 0) := "0000";

begin

process(CLKK)

begin

if (CLKK'EVENT) and (CLKK = '1') **then**

if(M = '0') **then** Y <= Y + "0001";

else Y <= Y - "0001"; **end if**;

end if;

end process;

 Q <= Y;

end rnr4;

Acest numarator realizeaza trecerea de la un etaj la altul. In functie de sensul de deplasare (M=0 pentru urcare si M=1 pentru coborare) acesta va numara crescator sau descrescator intre 0-12.

10. Divizor de frecventa pentru afisor

entity divAfis **is**

port (CLK: **in** std_logic;

D: **out** std_logic_vector(2 **downto** 0));

end divAfis;

architecture AdivAfis **of** divAfis **is**

signal S : std_logic_vector (18 **downto** 0):= "00000000000000000000";

begin

process (CLK,S)

begin

if (CLK'EVENT) and (CLK='1') **then** S<= S+1;

end if;

D(0)<= S(16);

D(1)<= S(17);

D(2)<= S(18);

end process;

end AdivAfis;

Divizeaza frecventa semnalului de tact al placii pentru a obtine o rata de refresh potrivita afisorului.

11. Decodificator 7 segmente

entity E_7segm **is**

port (binar:**in** std_logic_vector (3 **downto** 0);

seg0,seg1: **out** std_logic_vector (0 **to** 6));

end E_7segm;

architecture AE_7segm **of** E_7segm **is**

begin

process (binar)

begin

case binar **is**

when "0000" => seg0 <= "0000001"; --0

 seg1<="1111111";

when "0001" => seg0 <= "1001111"; --1

 seg1<="1111111";

when "0010" => seg0 <= "0010010"; --2

 seg1<="1111111";

when "0011" => seg0 <= "0000110"; --3

 seg1<="1111111";

when "0100" => seg0 <= "1001100"; --4

 seg1<="1111111";

when "0101" => seg0 <= "0100100"; --5

 seg1<="1111111";

when "0110" => seg0 <= "0100000"; --6

 seg1<="1111111";

when "0111" => seg0 <= "0001111"; --7

 seg1<="1111111";

when "1000" => seg0 <= "0000000"; --8

 seg1<="1111111";

when "1001" => seg0 <= "0000100"; --9

 seg1<="1111111";

when "1010" => seg0 <= "0000001"; --10

 seg1<="1001111";

when "1011" => seg0 <="1001111"; --11

 seg1<="1001111";

when "1100" => seg0 <="0010010"; --12

 seg1<="1001111";

when others => seg0 <= "1111111";

```

    seg1<="1111111";

    end case;

end process;

end AE_7segm;

```

Decodifica un numar in binar reprezentat pe 4 biti pentru a putea fi afisat.

12. Usa deschisa

entity usad **is**

```

    port(

        nr6: in std_logic_vector(2 downto 0);
        nr8: in std_logic_vector(3 downto 0);
        SU,SG,opreste,viteza: in std_logic;
        usadeschisa: out std_logic);

```

end usad;

architecture ausad **of** usad **is**

begin

```

    process(nr6,nr8,SG,SU,opreste)

```

```

        variable o:std_logic:='0';

```

```

        begin

```

```

        if(opreste = '1') then

```

```

            if(viteza = '0') then

```

```

                if ((nr6<"101") and (nr6>"001")) or (SG = '0') or (SU = '0') then o:='1';

```

```

                else o:='0';end if;

```

```

            else

```

```

                if ((nr8<"101") and (nr8>"001")) or (SG = '0') or (SU = '0') then o:='1';

```

```

                else o:='0';end if;

```

```

            end if;

```

```

        else o:='0';

```

```

        end if;

```

```

        usadeschisa<=o;

```

```

end process;

end ausad;

```

Folosind starea numaratoarelor si a senzorilor stabileste daca usa este deschisa sau nu.

13. Afisor

```

entity afisor is

    port (CLK : in std_logic_vector(2 downto 0);
          NR: in std_logic_vector(3 downto 0);
          sens,opreste,repas,SU,SG,D,viteza: in std_logic;
          COM: in std_logic_vector(3 downto 0);
          ANOD: out std_logic_vector(7 downto 0);
          CATOD: out std_logic_vector(0 to 6));

end afisor;

```

architecture Aafisor of afisor is

```

component E_7segm is

    port (binar:in std_logic_vector (3 downto 0);
          seg0,seg1: out std_logic_vector (0 to 6));

end component;

```

```

signal  A0 :std_logic_vector(0 to 6);
signal  A1 :std_logic_vector(0 to 6);
signal  C0 :std_logic_vector(0 to 6);
signal  C1 :std_logic_vector(0 to 6);

```

begin

```

E0: E_7segm port map(NR,A0,A1);
E1: E_7segm port map(COM,C0,C1);

process (CLK,A0,A1)

begin

```

```

if (CLK = "000") then ANOD <= "11111110"; CATOD <= A0;
elsif (CLK = "001") then ANOD<="11111101"; CATOD <= A1;
elsif (CLK = "010") then ANOD<="11101111"; CATOD <= C0;
elsif (CLK = "011") then ANOD<="11011111"; CATOD <= C1;
elsif (CLK = "100") then ANOD<="10111111";

    if (opreste = '1') then

        if (SG = '0') or (SU = '0') then

            CATOD<="0110000";

        else

            CATOD<="11111111";

        end if;

    else CATOD<="11111111";

    end if;

elsif (CLK = "101") then ANOD<="11111011";

    if(D = '1') or (repaus = '0') then CATOD<="1100011";

    else CATOD<="11111111"; end if;

elsif (CLK = "110") then ANOD<="01111111";

    if(viteza = '0') then CATOD<="1001111";

    else CATOD<="0000110"; end if;

else ANOD<="11110111";

    if(sens = '0') then CATOD<="1000001";

    else CATOD<="0110001"; end if;

    end if;

end process;

end Aafisor;

```

Afiseaza etajul curent, viteza si sensul de deplasare, daca usile sunt deschise, eroare daca unul dintre senzori are valoare 0, si comanda pentru a facilita introducerea ei.

4.3 Lift (top-level)

```

library IEEE;

use IEEE.STD_LOGIC_1164.all;

use IEEE.STD_LOGIC_UNSIGNED.all;

use IEEE.STD_LOGIC_ARITH.all;

entity lift is

    port (Viteza,CLK,SG,SU,OK,OKU,OKC: in std_logic;
          cerere: in std_logic_vector (3 downto 0);
          ANOD: out std_logic_vector(7 downto 0);
          CATOD: out std_logic_vector(0 to 6);
          repaus: out std_logic;
          opreste: out std_logic);

end lift;

```

architecture Alift of lift is

```

component rnum4 is

    port(CLKK,M: in std_logic;
          Q: out std_logic_vector (3 downto 0));

end component;

```

```

component mux41 is

    port (I0,I1,I2,I3,S0,S1:in std_logic;
          Q: out std_logic);

end component;

```

```

component si3 is

    port(A,B,C: in std_logic;
          Y: out std_logic);

end component;

```

```

component num3 is

    port (CLK,RESET: in std_logic;
          Q: out std_logic);

```

end component;

component num6 is

port (CLK,RESET: in std_logic;

Q: out std_logic;

nr: out std_logic_vector(2 downto 0));

end component;

component si2 is

port(A,B: in std_logic;

Y: out std_logic);

end component;

component UC is

port (

OK,OKU,OKC,CLK:in std_logic;

NR: in std_logic_vector (3 downto 0);

COM: in std_logic_vector (3 downto 0);

sens,opreste,repas: out std_logic);

end component;

component afisor is

port (CLK : in std_logic_vector(2 downto 0);

NR: in std_logic_vector(3 downto 0);

sens,opreste,repas,SG,SU,D,viteza: in std_logic;

COM: in std_logic_vector(3 downto 0);

ANOD: out std_logic_vector(7 downto 0);

CATOD: out std_logic_vector(0 to 6));

end component;

component divAfis is

port (CLK : in std_logic;

D : out std_logic_vector(2 downto 0));

end component;

component div1 **is**

port (CLK : **in** std_logic;

 D : **out** std_logic);

end component;

component num8 **is**

port (CLK,RESET: **in** std_logic;

 Q: **out** std_logic;

 nr: **out** std_logic_vector(3 **downto** 0));

end component;

component usad **is**

port(

 nr6: **in** std_logic_vector(2 **downto** 0);

 nr8: **in** std_logic_vector(3 **downto** 0);

 SU,SG,opreste,viteza: **in** std_logic;

 usadeschisa: **out** std_logic);

end component;

signal S1: std_logic;

signal S2: std_logic;

signal S3: std_logic;

signal S4: std_logic;

signal S5: std_logic;

signal S6: std_logic;

signal S7: std_logic;

signal S8: std_logic;

signal S9: std_logic := '0';

signal S10: std_logic := '0';

signal S11: std_logic := '0';

signal S12: std_logic;

signal Et: std_logic_vector (3 **downto** 0) := "0000";

signal CLK1: std_logic;

signal CLK_afisor: std_logic_vector(2 **downto** 0);

```

signal nr6: std_logic_vector(2 downto 0);
signal nr8: std_logic_vector(3 downto 0);
signal D: std_logic;
begin
    E0: div1 port map (CLK,CLK1);
    E1: si2 port map(CLK1,S11,S8);
    E2: num8 port map(S8,S9,S1,nr8);
    E3: num6 port map(S8,S9,S12,nr6);
    E4: si3 port map(SG,SU,S12,S3);
    E5: si3 port map(SG,SU,S1,S2);
    E6: num3 port map(S8,S5,S4);
    E7: mux41 port map(S8,S3,S4,S2,S9,Viteza,S5);
    E8: rnum4 port map(S5,S10,Et);
    E9: UC port map(OK,OKU,OKC,S5,Et,cerere,S10,S9,S11);
    opreste<=S9;
    repaus<=S11;
    E10: divAfis port map (CLK,CLK_afisor);
    E11: afisor port map (CLK_afisor,Et,S10,S9,S11,SG,SU,D,viteza,cerere,ANOD,CATOD);
    E12: usad port map(nr6,nr8,SU,SG,S9,viteza,D);
end architecture;

```

5. Justificarea solutiei alese

Am gandit si proiectat un algoritm eficient pentru functionarea liftului, atat din punct de vedere al timpului cat si din punct de vedere al resurselor utilizate.

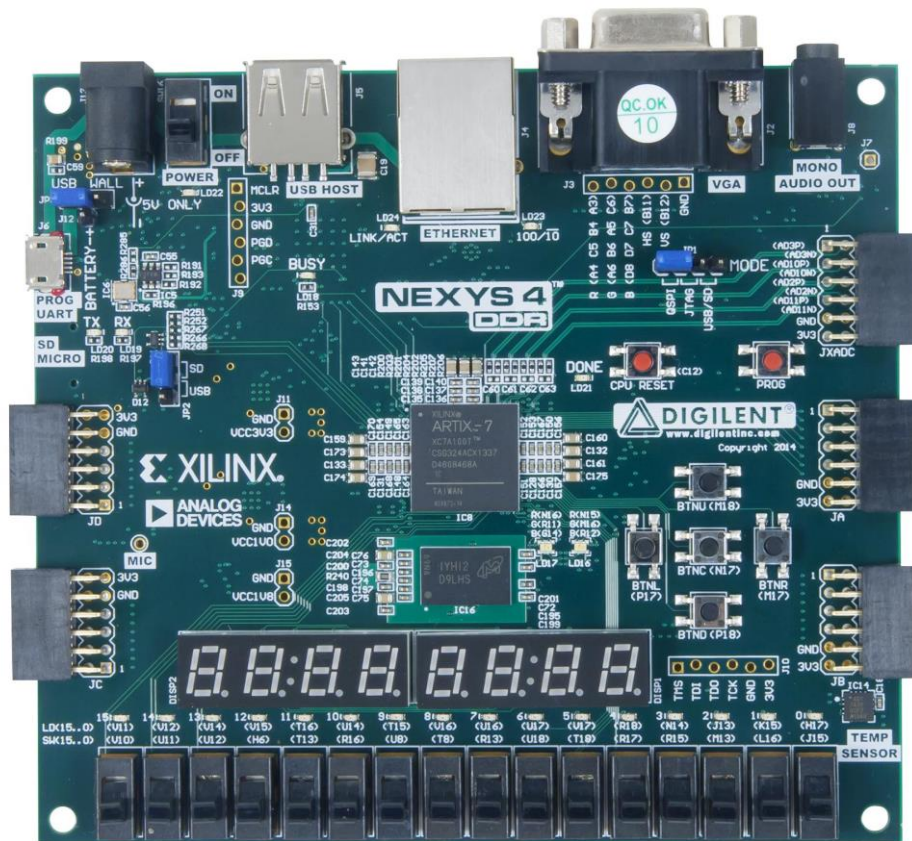
Am ales ca liftul sa se intoarca la parter cand nu mai sunt cereri, deoarece intr-un hotel deplasarea se face, in general, intre parter si unul din etaje.

Metoda noastra consta in impartirea proiectului pe „bucati”, fiecare avand functii diferite ,indeplinindu-si rolul in proiect. Acestea sunt legate in modulul principal printr-o descriere structurala, reprezentand rezultatul cerintei .

6. Instrucțiuni de utilizare

Pentru a putea programa proiectul pe placa trebuie să respectăm următorii pași:

1. Se deschide programul Xilinx ISE
2. Se creează un Workspace
3. Se face click pe "User constraints" , iar apoi se selectează "Edit constraints" ,unde se declară legăturile
4. Se da click pe "Generate Programming File"
5. Se da click pe "Configure Target Device"
5. Se da click pe "Boundary Scan"
6. Se da click pe „Initialize Chain"
7. Se caută programul făcând click pe „Browse" și apoi căutându-se fișierul cu extensia **.bit**
8. Se da click pe „Program" și urmărește evoluția pe FPGA



Switch-uri:

"Viteza" LOC=V10 - 0 pentru 1 secunda , 1 pentru 3 secunde

"SG" LOC=U11 - Senzor Greutate

"SU" LOC=U12 - Senzor Usa

"cerere(0)" LOC=T8

"cerere(1)" LOC=U8

"cerere(2)" LOC=R16

"cerere(3)" LOC=T13

Butoane:

"OK" LOC=N17 - Inregistrare cerere interior

"OKU" LOC=M18 - Inregistrare cerere exterior, urcare

"OKC" LOC=P18 - Inregistrare cerere exterior, coborare



7. Posibilitati de dezvoltare

Adaugarea unor semnale sonore atunci cand usile se inchid si deschid. Afisarea timpului in care liftul o sa ajunga la un anumit etaj. Muzica de fundal.

