

Automat bilete de tren

Grupa: 30216

Seria B



FACULTATEA: Automatica si Calculatoare

SPECIALIZAREA: Calculatoare si Tehnologia informatiei

DISCIPLINA: Proiectarea dispozitivelor numerice

PROIECT: Automat bilete de tren

Indrumator Laborator:

Lucia Vacariu

Realizator:

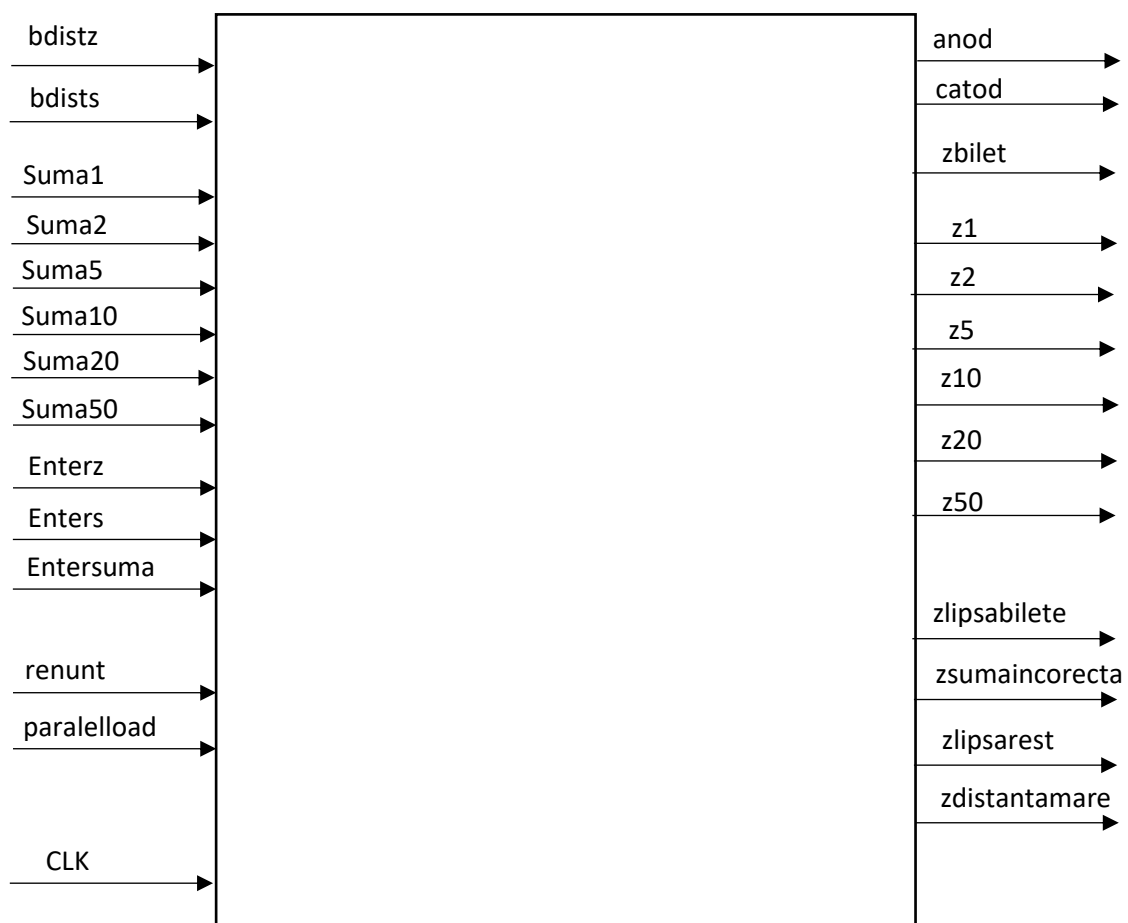
Ionela Fagadar

1.Specificatia proiectului

Tema proiectului: Sa se proiecteze un automat pentru cumpararea biletelor de tren. Cumparatorul introduce distanta pana la destinatie (in zeci de km). Costul biletului si sumele introduse sunt afisate pe afisoare 7 segmente. Moneda utilizata este EURO. Pretul maxim pentru un bilet este de 100 Euro. Automatul primeste suma necesara in hartii sau monede si elibereaza biletele si, eventual, restul. El dispune de o casa de bani care se incarca la inceputul functionarii cu un numar de hartii si monede (toate posibilitatile intre 1 euro si 50 euro). Lipsa de bilete,introducerea unei sume mai mici decat costul biletului sau imposibilitatea restituirii restului se semnalizeaza luminous. Se poate renunta in orice moment la operatie, cu restituirea sumei introduse, daca este cazul.

2.Schema bloc

Intai, automatul va fi prezentat sub forma unei cutii negre, cu intrarile si iesirile sale:



Iesiri:

-anod,catod: Iesirile pentru afisorul 7 segmente, care afiseaza suma ceruta si suma introdusa.

-zbilet: Led care arata cand se emite biletul.

-z1,z2,z5,z10,z20,z50: Leduri care arata ce fel de bancnota emite automatul.

-zlipsabilete: Led care semnalizeaza luminos lipsa biletelor.

-zsumaincorecta: Led care semnalizeaza luminos ca suma introdusa nu este cea ceruta, ci este mai mica

-zlipsarest: Led ce semnalizeaza imposibilitatea restituirii restului.

-zdistantamare: Led ce semnalizeaza ca distanta introdusa este mai mare de 100 km.

Componente:

-blocul afisor: Bloc format din cele 8 afisoare pe 7 segmente, care afiseaza suma ceruta si cea introdusa. Intrari:suma introdusa, cost bilet,CLK. Iesiri: Anod,catod.

-calcul distanta:Aceasta componenta are intrarile bdistz,bdists,enterz,enters,enablesum si calculeaza distanta introdusa, verificand daca este indeplinita conditia: $\text{distanta} \leq 100 \text{ km}$

-cost bilet: Calculeaza cat este costul biletului, in functie de distanta introdusa. Intrari:distanta introdusa, enablesum , iar iesirea este costul biletului. (Am stabilit ca 1 km = 1 Euro)

-calcul rest: Calculeaza restul pe care trebuie sa il restituie automatul, cand enabledif este activ. Primeste costul biletului, suma introdusa(sumai) si verifica daca suma este corecta (altfel, zsumaincorecta se aprinde), calculand apoi restul.

-casa bani: Aceasta face tranzactiile, astfel, dupa ce a primit costul biletului, se introduc bancnotele(sumai, suma2, suma5, suma10, suma20, suma50, reprezentand cele 6 tipuri de bancnote acceptate de automat), pe rand, actualizand numarul bancnotelor din casa de bani. Aceasta mai are intrarile enable, enablesuma, enablerest, care fac diferenta intre momentul cand se introduc bani, si cand se dau bani, resti, paralelload pentru fiecare tranzactie si CLK. Aceasta componenta calculeaza sumaintrodusa (sumai), da restul (z1,z2,z5, z10, z20, z50), biletul (zbilet), actualizandu-se si aici numarul bancnotelor de fiecare fel, cat si numarul biletelor.Se semnalizeaza luminos imposibilitatea restituirii restului(zlipsarest), cat si lipsa biletelor(zlipsabilete).

3.Proiectare si implementare

3.1 Descrierea schemei de detaliu

3.2 Proiectare componente

Comparator (pe 8 biti)

-Comparatorul compara doua numere, A si B, iar daca $A > B$, ymare devine 1, daca $A = B$, yegal devine 1, $A < B$, ymic devine 1. Aici, componenta compara distanta introdusa cu 100, iar daca este mai mare, se semnalizeaza printr-un led (primul comparator). De asemenea, se compara suma introdusa cu cea ceruta, iar daca suma introdusa este mai mica decat cea ceruta, se semnalizeaza luminos (al doilea comparator).

```
library ieee;
use ieee.std_logic_1164.all;

entity comparator is
    port (a,b:in std_logic_vector(7 downto 0);    --numere pe 8 biti
          ymic,yegal,ymare: out std_logic);        --rezultatul compararii
end entity;

architecture comparator_a of comparator is
begin
    yegal <= '1' when (a=b) else '0';              --egale
    ymic <= '1' when (a<b) else '0';               --a mai mic decat b
    ymare <= '1' when (a>b) else '0';              --a mai mare decat b
end architecture ;
```

Decodificatorz

-Decodificatorul transforma un numar de 4 biti, intr-unul de 8. Aici, decodifica informatia primita de la numaratorul plasat la butonul zecilor. (ex: 2 in binar la intrare, reprezinta de fapt 20, si devine 20 in binar la iesire)

```
library ieee;
use ieee.std_logic_1164.all;

entity decodificatorz is
    port( a: in std_logic_vector (3 downto 0);
          q :out std_logic_vector (7 downto 0));
end entity;

architecture Arh of decodificatorz is
begin
```

```

process(a)
begin
case a is
    when "0000"=>q<="00000000";      --0 zeci = 0
    when "0001"=>q<="00001010";      --1 zeci = 10
    when "0010"=>q<="00010100";      --2 zeci = 20
    when "0011"=>q<="00011110";      --3 zeci = 30
    when "0100"=>q<="00101000";      --4 zeci = 40
    when "0101"=>q<="00110010";      --5 zeci = 50
    when "0110"=>q<="00111100";      --6 zeci = 60
    when "0111"=>q<="01000110";      --7 zeci = 70
    when "1000"=>q<="01010000";      --8 zeci = 80
    when "1001"=>q<="01011010";      --9 zeci = 90
    when others=>q<="00000000";      --altii = 0

end case;
end process;
end architecture Arh;

```

Decodificators

-Decodificatorul transforma un numar de 4 biti, intr-unul de 8. Aici, decodifica informatia primita de la numaratorul plasat la butonul sutelor. (ex: 1 in binar la intrare, reprezinta de fapt 100, si devine 100 in binar la iesire)

```

library ieee;
use ieee.std_logic_1164.all;

entity decodificators is
    port( a: in std_logic_vector (3 downto 0);
          q :out std_logic_vector (7 downto 0));
end entity;

architecture Arh of decodificators is
begin
    process(a)
    begin
    case a is
        when "0000"=>q<="00000000";      --0 sute = 0
        when "0001"=>q<="01100100";      --1 sute = 100

```

```

        when others=>q<="01100101";

--de aici, nu mai conteaza, pentru ca, distanta trebuie sa fie mai mica decat 100

        end case;

    end process;

end architecture Arh;

```

Convertor binar bcd (binarbcd)

-converteste codul din binar in bcd, pentru a putea fi afisat pe afisoare 7 segmente. Se foloseste algoritmul „ADUNA 3”

```

library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;

entity binarbcd is

    port( binar: in std_logic_vector (7 downto 0); --nr in binar ce trebuie convertit (8 biti)

          bcd_s: out std_logic_vector (3 downto 0);      --sutele nr

          bcd_z: out std_logic_vector (3 downto 0);      --zecile nr

          bcd_u: out std_logic_vector (3 downto 0));      --unitatile nr

end entity binarbcd;

architecture binarbcd_a of binarbcd is

begin

    process (binar)

        variable binar_var: std_logic_vector (7 downto 0);

        variable bcd :std_logic_vector (11 downto 0);

    begin

        binar_var := binar;

        bcd := "000000000000";

        for i in 0 to 7 loop                                --algoritmul aduna 3

            if bcd(3 downto 0) > "0100" then

                bcd(3 downto 0) := bcd(3 downto 0) + "0011";

            end if;

            if bcd(7 downto 4) > "0100" then

                bcd(7 downto 4) := bcd(7 downto 4) + "0011";

            end if;

        end loop;

    end process;

end architecture binarbcd_a;

```



```

        if bcd(11 downto 8) > "0100" then
            bcd(11 downto 8) := bcd(11 downto 8) + "0011";
        end if;

        bcd := bcd(10 downto 0) & binar_var(7);
        binar_var := binar_var(6 downto 0) & '0';

    end loop;

    bcd_s <= bcd(11 downto 8);
    bcd_z <= bcd(7 downto 4);
    bcd_u <= bcd(3 downto 0);

end process;
end architecture binarbcd_a;

```

Numarator reversibil (numaratorrev)

-numara de cate ori s-a apasat pe butonul destinat sutelor si zecilor de km. Acesta are un CLK, amplasat la clock-ul placutei FPGA, si un enable, legat la butoane, pentru a incrementa valoarea doar atunci cand butonul este apasat.

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity numaratorrev is
    port( a: in std_logic_vector (3 downto 0);
          clock :in std_logic;
          sens :in std_logic;          --'0' pt descrescator, '1' pt crescator
          PL : in std_logic;          --incarcare paralela
          enable: in std_logic;        --enable dispozitiv
          enter: in std_logic;         --enter suma
          y:out std_logic_vector(3 downto 0));
end entity numaratorrev;

architecture numaratorrev_a of numaratorrev is
    signal Q:std_logic_vector (3 downto 0);

    begin
        process(clock,sens,PL)
            begin

```

```

    if PL = '1' then Q<=a;      --incarcare paralela
    end if;

    if (clock='1' and clock'EVENT and PL='0' and enter='0') then

        if (Q/= "0000" and sens='0' and enable='1') then

            Q<=Q-1;              --numarare descrescatoare

        elsif (Q/= "1010" and sens='1' and enable='1') then

            Q<=Q+1;              --numarare crescatoare

        end if;

        if Q="1010" then Q<="0000";    --numaratorul numara pana la 9

        end if;

    end if;

end process;

y<= Q;

end architecture numaratorrev_a;

```

Debounce

-folosita pentru debounce-ul butoanelor sau a switch-urilor, deoarece, clock-ul placutei FPGA este foarte rapid si astfel se vor percepe prea repede valorile date intrarilor.

```

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity debounce is

    port(input: in std_logic;

        clock :in std_logic;

        output:out std_logic);

end entity;

architecture debounce_a of debounce is

    signal r1,r2,r3:std_logic;

    signal en:std_logic_vector(15 downto 0):=(others =>'0');

begin

    process(clock) is

    begin

        if rising_edge(clock) then

            en<=en+1;

            if en=0 then

```

```

                r1<=input;

            end if;

            r2<=r1;

            r3<=r2;

        end if;

    end process;

    output<=r2 and (not r3);

end architecture;

```

Sumator scazator

-Sumatorul scazator este o componenta utilizata pentru a aduna doua numere A si B, daca suma='1' si pentru a scadea, daca suma='0'. Functia de sumator este folosita pentru a calcula distanta introdusa, iar cea de scazator, pentru rest.

```

library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sumatorscazator is

    port(A:in std_logic_vector(7 downto 0);

        B: in std_logic_vector(7 downto 0);

        enable,suma:in std_logic; --daca suma ='1' se aduna, altfel se scade

        Y: out std_logic_vector(7 downto 0));

end entity;

architecture arh of sumatorscazator is

begin

    process(enable)

    begin

        if enable='1' then

            if(suma='1') then

                Y<=A+B;          --se aduna

            else

                Y<=A-B;          --se scade

            end if;

        end if;

    end process;

end architecture;

```

```
end architecture arh;
```

Afisor 7 segmente (seven_segment)

-folosit pentru a afisa suma introdusa si suma ceruta. Acesta este proiectat astfel incat, un singur afisor din cele 8 functioneaza, intr-un semnal de tact. Acesta este format dintr-un decodificator, care decodifica intrarile. Cu ajutorul divizorului de frecventa, afisorul este programat astfel incat, ochiul uman sa nu mai perceapa faptul ca afisoarele se aprind unul dupa altul, ci, la o astfel de viteza, sa perceapa ca sunt aprinse simultan.

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.std_logic_unsigned.all;
```

```
entity seven_segment is
```

```
Port ( clock: in STD_LOGIC;    --clock de 100Mhz
```

```
      nr1,nr2,nr3,nr4,nr5,nr6,nr7,nr8:in STD_LOGIC_VECTOR (3 downto 0); --nr de afisat
```

```
      anod : out STD_LOGIC_VECTOR (7 downto 0);
```

```
      Catod : out STD_LOGIC_VECTOR (6 downto 0));
```

```
end seven_segment;
```

```
architecture arh of seven_segment is
```

```
signal nr_bcd: STD_LOGIC_VECTOR (3 downto 0);
```

```
signal refresh_counter: STD_LOGIC_VECTOR (19 downto 0);
```

```
signal activ: std_logic_vector(2 downto 0);
```

```
begin
```

```
process(nr_bcd,clock) --decodificator
```

```
begin
```

```
  case nr_bcd is
```

```
    when "0000" => Catod <= "0000001"; -- "0"
```

```
    when "0001" => Catod <= "1001111"; -- "1"
```

```
    when "0010" => Catod <= "0010010"; -- "2"
```

```
    when "0011" => Catod <= "0000110"; -- "3"
```

```
    when "0100" => Catod <= "1001100"; -- "4"
```

```
    when "0101" => Catod <= "0100100"; -- "5"
```

```
    when "0110" => Catod <= "0100000"; -- "6"
```

```
    when "0111" => Catod <= "0001111"; -- "7"
```

```
    when "1000" => Catod <= "0000000"; -- "8"
```

```
    when "1001" => Catod <= "0000100"; -- "9"
```

```
    when others => Catod <= "1111111";
```

```
    end case;
end process;
process(clock) --divizor de frecventa
begin
    if(rising_edge(clock)) then
        refresh_counter <= refresh_counter + 1;
    end if;
end process;
activ <= refresh_counter(19 downto 17);
process(activ,nr1,nr2,nr3,nr4) --aprindere afisoare in functie de anod
begin
    case activ is
        when "000" =>
            anod <= "01111111";
            -- activeaza ledul 1 si ii dezactiveaza pe ceilalti
            nr_bcd <= nr1;
        when "001" =>
            anod <= "10111111";
            -- activeaza ledul 2 si ii dezactiveaza pe ceilalti
            nr_bcd <= nr2;
        when "010" =>
            anod <= "11011111";
            -- activeaza ledul 3 si ii dezactiveaza pe ceilalti
            nr_bcd <= nr3;
        when "011" =>
            anod <= "11101111";
            -- activeaza ledul 4 si ii dezactiveaza pe ceilalti
            nr_bcd <= nr4;
            when "100" =>
                anod <= "11110111";
                -- activeaza ledul 5 si ii dezactiveaza pe ceilalti
                nr_bcd <= nr5;
            when "101" =>
                anod <= "11111011";
                -- activeaza ledul 6 si ii dezactiveaza pe ceilalti
                nr_bcd <= nr6;
```

```
        when "110" =>
            anod <= "11111101";
            -- activeaza ledul 7 si ii dezactiveaza pe ceilalti
        nr_bcd <= nr7;
        when "111" =>
            anod <= "11111110";
            -- activeaza ledul 8 si ii dezactiveaza pe ceilalti
        nr_bcd <= nr8;
        when others =>
            anod <= "11111111";
            --dezactieaza tot
        nr_bcd <= nr8;
    end case;
end process;
end arh;
```

Casa bani

-folosita pentru a efectua tranzactiile, a da bilete si rest. Aceasta semnalizeaza lipsa biletelor si imposibilitatea restituirii restului.

library IEEE;

use IEEE.std_logic_1164.all;

use ieee.numeric_std.all;

entity casabani is

port(e1, e2, e5, e10, e20, e50:in std_logic;

clk:in std_logic;

resti:in std_logic_vector(7 downto 0);

paralelload:in std_logic;

enableintroducere,enablerest,enable:in std_logic;

error:in std_logic;

sumaintrodusaafis:out std_logic_vector(7 downto 0);

lipsarest,lipsabilete:out std_logic;

zbilet,z1,z2,z5,z10,z20,z50: out std_logic);

end entity casabani;

architecture arh of casabani is

```
signal suma_introdusa: std_logic_vector(7 downto 0) := "00000000";
```

```
signal refresh_counter:integer:=0;
```

```
begin
```

```
process(clk,e1,e2,e5,e10,e20,e50,paralelload,enableintroducere,enable)
```

```
variable bani1,bani2,bani5,bani10,bani20,bani50:natural range 0 to 10 :=2;
```

```
variable r,q:integer range 0 to 100;
```

```
variable casa_bilete:std_logic_vector(7 downto 0):="00000010";
```

```
variable error_rest:std_logic := '0';
```

```
variable error_bilet:std_logic := '0';
```

```
variable da_bani:std_logic:='0';
```

```
variable baniireturnati:std_logic_vector(7 downto 0):="00000000";
```

```
begin
```

```
if paralelload = '1' then      --incarcare paralela
```

```
    lipsarest<='0';
```

```
    zbilet<='0';
```

```
    z1<='0';
```

```
    z2<='0';
```

```
    z5<='0';
```

```
    z10<='0';
```

```
    z20<='0';
```

```
    z50<='0';
```

```
    da_bani:='0';
```

```
    suma_introdusa<="00000000";
```

```
elsif clk='1' and clk'event and enable='1' then
```

```
    if casa_bilete="00000000" then      --daca nu mai sunt bilete, se semnalizeaza luminos
```

```
        lipsabilete<='1';
```

```
        error_bilet:='1';
```

```
    else lipsabilete<='0';
```

```
    end if;
```

```
    if enableintroducere='1' then
```

```
        --stare introducere bani
```

```
        if e1='1' then
```

```
            suma_introdusa <= std_logic_vector(unsigned(suma_introdusa) + 1);
```

```
            bani1 := bani1 + 1;
```

```
        elsif e2='1' then
```

```
            suma_introdusa <= std_logic_vector(unsigned(suma_introdusa) + 2);
```



```

    bani2 := bani2 + 1;
    elsif e5='1' then
        suma_introdusa <= std_logic_vector(unsigned(suma_introdusa) + 5);
    bani5 := bani5 + 1;
    elsif e10='1' then
        suma_introdusa <= std_logic_vector(unsigned(suma_introdusa) + 10);
    bani10 := bani10 + 1;
    elsif e20='1' then
        suma_introdusa <= std_logic_vector(unsigned(suma_introdusa) + 20);
    bani20 := bani20 + 1;
    elsif e50='1' then
        suma_introdusa <= std_logic_vector(unsigned(suma_introdusa) + 50);
    bani50 := bani50 + 1;
    end if;
end if;

if enablerest='1' and enableintroducere='0' then                --stare rest
    r := to_integer(unsigned(resti));
    if r /= 0 then                                                --bani50
        q := r/50;
        if bani50 >= q then r := r mod 50;
        else r := r - 50 * bani50;
        end if;
    end if;
    if r /= 0 then                                                --bani20
        q := r/20;
        if bani20 >= q then r := r mod 20;
        else r := r - 20*bani20;
        end if;
    end if;
    if r /= 0 then                                                --bani10
        q := r/10;
        if bani10 >= q then r := r mod 10;
        else r := r - 10*bani10;
        end if;
    end if;
    if r /= 0 then                                                --bani5

```

```

    q := r/5;
    if bani5 >= q then r := r mod 5;
        else r := r - 5*bani5;
    end if;
end if;
if r /= 0 then                                --bani2
    q := r/2;
    if bani2 >= q then r := r mod 2;
        else := r - 2*bani2;
    end if;
end if;
if r /= 0 then                                --bani1
    q := r/1;
    if bani1 >= q then r := r mod 1;
        else r := r - bani1;
    end if;
end if;
-- verificare rest
if r /= 0 then
    lipsarest <= '1';
    error_rest := '1';
else
    lipsarest <= '0';
    error_rest := '0';
end if;
if r=0 and da_bani='0' and error='0' and error_rest='0' and error_bilet='0' then
    baniireturnati := resti;
    da_bani := '1';
    casa_bilete:= std_logic_vector(unsigned(casa_bilete) - 1);
    zbilet<='1';                                --daca nu exista eroare, se da bilet si se da restul
elsif r=0 and da_bani='0' and ( error='1' or error_rest='1' or error_bilet='1') then
    baniireturnati :=suma_introdusa;
    da_bani := '1';
    casa_bilete:= std_logic_vector(unsigned(casa_bilete));
    zbilet<='0';                                --daca exista eroare, nu se da bilet si se restituie suma initiala
end if;

```

```
end if;

if da_bani = '1' then                                --stare in care automatul returneaza banii
    if unsigned(baniireturnati) >= 0 then
        if(rising_edge(clk)) then
            refresh_counter <= refresh_counter + 1;
        end if;  --divizor de frecventa incorporat, pentru aprinderea succesiva a ledurilor
        if refresh_counter=100000000 then
            refresh_counter<=0;
            if unsigned(baniireturnati) >= 50 and bani50 /= 0 then
                baniireturnati := std_logic_vector(unsigned(baniireturnati) - 50);
                bani50 := bani50 - 1;
                z1<='0';
                z2<='0';
                z5<='0';
                z10<='0';
                z20<='0';
                z50<='1';
            elsif unsigned(baniireturnati) >= 20 and bani20 /= 0 then
                baniireturnati := std_logic_vector(unsigned(baniireturnati) - 20);
                bani20 := bani20 - 1;
                z1<='0';
                z2<='0';
                z5<='0';
                z10<='0';
                z20<='1';
                z50<='0';
            elsif unsigned(baniireturnati) >= 10 and bani10 /= 0 then
                baniireturnati := std_logic_vector(unsigned(baniireturnati) - 10);
                bani10 := bani10 - 1;
                z1<='0';
                z2<='0';
                z5<='0';
                z10<='1';
                z20<='0';
                z50<='0';
            elsif unsigned(baniireturnati) >= 5 and bani5 /= 0 then
```

```
        baniireturnati := std_logic_vector(unsigned(baniireturnati) - 5);

        bani5 := bani5 - 1;
        z1<='0';
        z2<='0';
        z5<='1';
        z10<='0';
        z20<='0';
        z50<='0';

    elsif unsigned(baniireturnati) >= 2 and bani2 /= 0 then

        baniireturnati := std_logic_vector(unsigned(baniireturnati) - 2);
        bani2 := bani2 - 1;
        z1<='0';
        z2<='1';
        z5<='0';
        z10<='0';
        z20<='0';
        z50<='0';

    elsif unsigned(baniireturnati) >= 1 and bani1 /= 0 then

        baniireturnati := std_logic_vector(unsigned(baniireturnati) - 1);
        bani1 := bani1 - 1;
        z1<='1';
        z2<='0';
        z5<='0';
        z10<='0';
        z20<='0';
        z50<='0';

    elsif baniireturnati = "00000000" then

        z1<='0';
        z2<='0';
        z5<='0';
        z10<='0';
        z20<='0';
        z50<='0';

    end if;

end if;

end if;
```

```

end if;
end if;
sumaintrodusaafis<=suma_introdusa;
end process;
end architecture ;

```

3.3 Proiectare de ansamblu

-In arhitectura top-level se interconecteaza elementele deja prezentate,astfel incat sa functioneze ca automatul descris.Aceasta are o structura mixta, deoarece se folosesc si procese pentru a sari dintr-o stare in alta, cat si pentru a semnaliza luminos.

```

library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.std_logic_arith.all;

use IEEE.std_logic_unsigned.all;

use std.standard.all;

entity main is

    port(bdistz,bdists: in std_logic;           --butoane

        suma1,suma2,suma5,suma10,suma20,suma50: in std_logic; --swich

        enterz,enters,entersuma:in std_logic;    --switch

        renunt:in std_logic;                     --switch

        CLK:in std_logic;                       --clock placuta FPGA

        anod: out std_logic_vector(7 downto 0);  --anod afisor 7 segmente

        catod: out std_logic_vector(6 downto 0); --catod afisor 7 segmente

        zbilet,z1,z2,z5,z10,z20,z50:out std_logic; --leduri pentru bilete si bancnote restituite

        zlipsabilete,zsumaincorecta,zlipsarest,zdistantamare:out std_logic; --leduri avertizare

        parallelload:in std_logic);             --switch

end entity main;

architecture arhitectura of main is

    component comparator

        port (a,b:in std_logic_vector(7 downto 0);

            ymic,yegal,ymare: out std_logic);

    end component;

    component decodicatorz

        port( a: in std_logic_vector (3 downto 0);

            q :out std_logic_vector (7 downto 0));

    end component;

    component decodificators

```

```
        port( a: in std_logic_vector (3 downto 0);
              q :out std_logic_vector (7 downto 0));
    end component;

    component binarbcd
        port( binar: in std_logic_vector (7 downto 0);
              bcd_s: out std_logic_vector (3 downto 0);
              bcd_z: out std_logic_vector (3 downto 0);
              bcd_u: out std_logic_vector (3 downto 0));
    end component;

    component numaratorrev
        port( a: in std_logic_vector (3 downto 0);
              clock :in std_logic;
              sens :in std_logic;
              PL : in std_logic;
              enable: in std_logic;
              enter: in std_logic;
              y:out std_logic_vector(3 downto 0));
    end component;

    component debounce
        port(input: in std_logic;
              clock :in std_logic;
              output:out std_logic);
    end component;

    component sumatorscazator
        port(A:in std_logic_vector(7 downto 0);
              B: in std_logic_vector(7 downto 0);
              enable,suma:in std_logic;
              Y: out std_logic_vector(7 downto 0));
    end component;

    component seven_segment
        Port ( clock: in STD_LOGIC;
              nr1,nr2,nr3,nr4,nr5,nr6,nr7,nr8:in STD_LOGIC_VECTOR (3 downto 0);
              anod : out STD_LOGIC_VECTOR (7 downto 0);
              Catod : out STD_LOGIC_VECTOR (6 downto 0));
    end component;
```

component casabani

```
port(e1, e2, e5, e10, e20, e50:in std_logic;
clk:in std_logic;
resti:in std_logic_vector(7 downto 0);
parallelload:in std_logic;
enableintroducere,enablerest,enable:in std_logic;
error:in std_logic;
sumaintrodusaafis:out std_logic_vector(7 downto 0);
lipsarest,lipsabilete:out std_logic;
zbilet,z1,z2,z5,z10,z20,z50: out std_logic));
```

end component;

```
signal nrzeci,nrsute:std_logic_vector(3 downto 0);
signal y2,y3:std_logic_vector(7 downto 0);
signal enablesum:std_logic;
signal distantaintrodusa:std_logic_vector(7 downto 0):="00000000";
signal ymic,yegal,ymare,y1mic,y1egal,y1mare:std_logic;
signal enablesuma:std_logic:='1';
signal enablerest,y:std_logic:='0';
signal bdistz_d,bdists_d,buton_reset_d:std_logic;
signal resti,sumaintrodusaafisata:std_logic_vector(7 downto 0):="00000000";
signal sumas,sumaz,sumau:std_logic_vector(3 downto 0);
signal enabledif:std_logic:='0';
signal suma1_d,suma2_d,suma5_d,suma10_d,suma20_d,suma50_d:std_logic;
begin
c0:debounce port map(bdistz,CLK,bdistz_d);    --debounce butoane
c1:debounce port map(bdists,CLK,bdists_d);    --debounce butoane
c2: numaratorrev port map("0000",CLK,'1',parallelload,bdistz_d,nrzeci);
c3: numaratorrev port map("0000",CLK,'1',parallelload,bdists_d,nrsute);
c4: decodicatorz port map(nrzeci,y2);          --decodificare informatie
c5: decodicators port map(nrsute,y3);          --decodificare informatie
c6:sumatorscazator port map(y2,y3,enablesum,'1',distantaintrodusa);    --se aduna zecile cu
sutele,pentru a se obtine distanta
c7:comparator port map(distantaintrodusa,"01100100",ymic,yegal,ymare); --se compara distanta cu 100
zdistantamare<=ymare;
y<=ymic or yegal;
c8:comparator port map(sumaintrodusaafisata,distantaintrodusa,y1mic,y1egal,y1mare);
```



```

c9:debounce port map(sum1,CLK,suma1_d);           --debounce butoane
c10:debounce port map(sum2,CLK,suma2_d);          --debounce butoane
c11:debounce port map(sum5,CLK,suma5_d);          --debounce butoane
c12:debounce port map(sum10,CLK,suma10_d);         --debounce butoane
c13:debounce port map(sum20,CLK,suma20_d);         --debounce butoane
c14:debounce port map(sum50,CLK,suma50_d);         --debounce butoane

c15:sumatorscazator port map (sumaintrodusaafisata,distantaintrodusa,enabledif,'0',resti);  --calcul rest

casa:casabani port
map(sum1_d,suma2_d,suma5_d,suma10_d,suma20_d,suma50_d,CLK,resti,paralelload,enablesuma,enableres
t,y,renunt,sumaintrodusaafisata,zlipsarest,zlipsabilete,zbilet,z1,z2,z5,z10,z20,z50);

process(y1mic) is
begin
    if y1mic='1' then                --sumaintrodusa e mai mica decat sumaceruta
        zsumaincorecta<='1';
    else zsumaincorecta<='0';
    end if;
end process;

process(clk) is                    --proces pentru schimbare intre starile automatului
begin
    if clk='1' and clk'event then
        if entersuma='0' then
            enablesum<=enterz and enters;
            enablesuma<='1';
            enabledif<='0';
            enablerest<='0';
        else
            enablesuma<='0';
            enabledif<=(ymic or yegal)and(y1mare or y1egal);
            enablerest<='1';
            enablesum<='0';
        end if;
    end if;
end process;

c16: binarbcd port map(sumaintrodusaafisata,sumas,sumaz,sumau);                --convertire in bcd

afisare:seven_segment port map(CLK,nrsute,nrzeci,"0000","1111",sumas,sumaz,sumau,"1111",anod,catot);

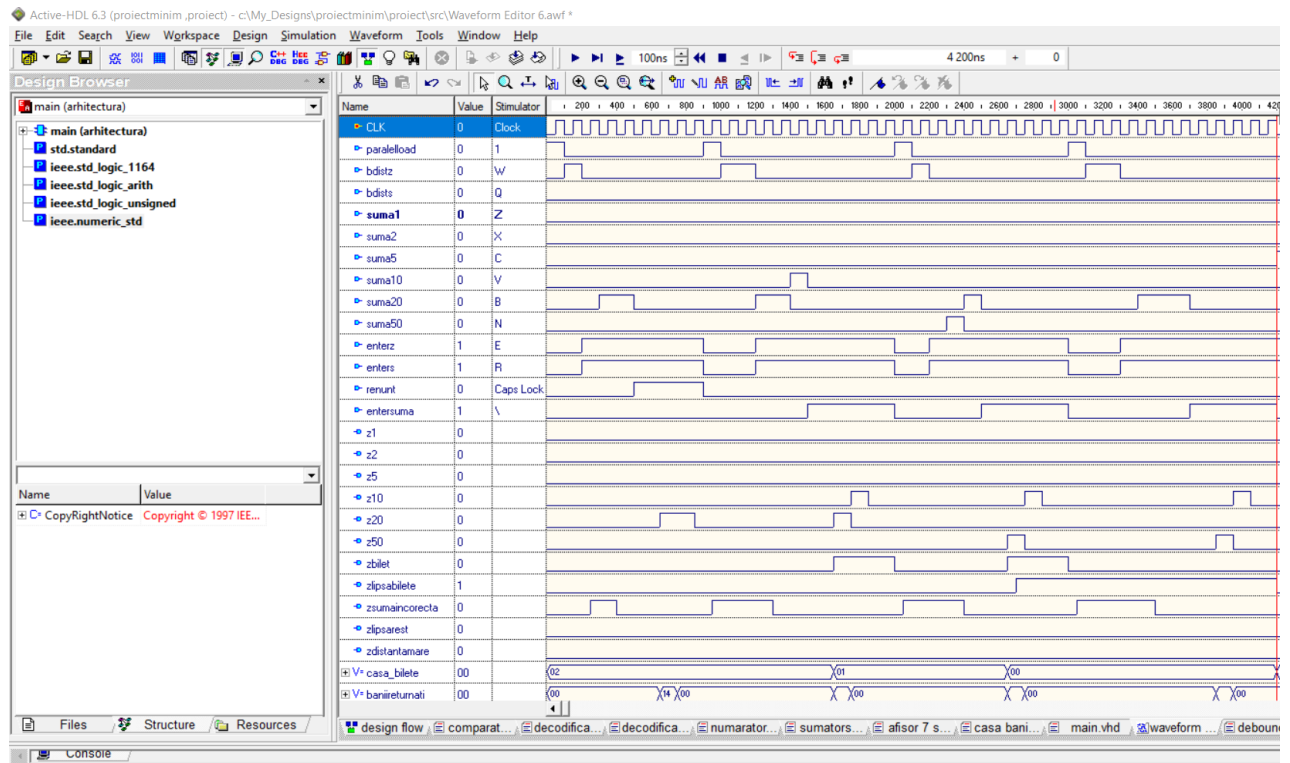
--afisare suma ceruta,suma introdusa

```

end architecture ;

3.4 Simulare in Active-HDL

-in exemplul urmatoar este prezentat modul cum automatul restituie banii, atunci cand operatia decurge in mod normal, atunci cand se renunta la tranzactie, cu restituirea sumei initiale, cat si atunci cand automatul nu mai dispune de bilete.



4. Lista componente utilizate

1. Comparator
2. Decodificatorz
3. Decodificators
4. Convertor binar bcd
5. Numarator reversibil
6. Debounce
7. Sumatorscazator
8. Afisor 7 segmente
9. Casa bani

5.Semnificatia notatiilor de intrare,iesire si a celor interne

- **Intrare:**

Bdistz,Bdists : zecile, respectiv sutele distantei

Suma1, suma2, suma5, suma10, suma20, suma50: bancnotele introduse

Enters, enterz: confirmarea introducerii sutelor si zecilor

Renunt: operatia de renuntare a tranzactiei

CLK: semnalul de tact

Parallelload: incarcare paralela (se utilizeaza la inceputul fiecărei tranzactii)

- **Iesire:**

Anod,Catod: specifice afisorului 7 segmente

Zbilet: se returneaza biletul

Z1, z2, z5, z10, z20, z50: se restituie restul in bancnote

Zsumaincorecta : se semnalizeaza luminos faptul ca suma introdusa nu este cea ceruta

Zlipsarest: se semnalizeaza luminos faptul ca automatul nu poate restitui restul

Zdistantamare: se semnalizeaza luminos ca distanta este mai mare de 100 km

- **Intermediare:**

Nrsute, Nrzeci: rezultatele returnate de numaratoarele reversibile(face legatura intre numarators/numaratorz si decoficators/decodificatorz, cat si intre numarators/numaratorz si afisor)

Y2,y3:numerele decodificate in binar(face legatura intre decodificatorz/decodificators si sumatorscazator)

Enablesum: enable pentru sumatorscazator 1

Distantaintrodusa: distanta introdusa de utilizator (face legatura intre sumatorscazator si cele 2 comparatoare, cat si intre al doilea sumatorscazator)

Ymic,yegal,ymare,y1mic,y1legal,y1mare: rezultatele comparatoarelor

Y: este un sau intre ymic si y egal (face legatura intre comparatorul 1 si casa de bani)

Enablesuma: stare pentru casa de bani

Enablerest: stare pentru casa de bani

Enabledif: enable pentru sumatorscazator 2

Bdistz_d, bdists_d: bdistz,bdists dupa debounce

Resti: restul calculat (face legatura intre sumatorscazator 2 si casa de bani)

Sumaintrodusaafisata: suma ce trebuie afisata (face legatura intre casa de bani si sumatorscazator 2, comparator 2 si binar bcd)

Sumas,sumaz,sumau: suma codificata in bcd (face legatura intre binar bcd si afisor)

suma1_d, suma2_d, suma5_d, suma10_d, suma20_d, suma50_d: Suma1, suma2, suma5, suma10, suma20, suma50 dupa debounce

6. Justificarea solutiei alese

Am ales aceasta solutie deoarece este cea mai simpla metoda posibila. Automatul este impartit in componente, care, sunt legate in arhitectura top-level, printr-o descriere mixta, deoarece, se folosesc atat componente, cat si procese prin care se trece dintr-o stare a automatului, in alta.

Automatul de bilete, este, in general, destul de utilizat in zilele noastre, deoarece minimizeaza timpul de asteptare, maximizand comoditatea.

7. Utilizare si rezultate:

7.1. Resurse necesare

Pentru utilizarea proiectului, este nevoie de un calculator, iar pentru etapa realizarii simularii in Active-HDL, este necesar sa fie instalat programul Active-HDL. Pentru etapa realizarii proiectului pe o placa FPGA, este necesar sa fie instalat programul ISE Foundation, iar calculatorul sa fie conectat la o placuta FPGA Nexys 4.

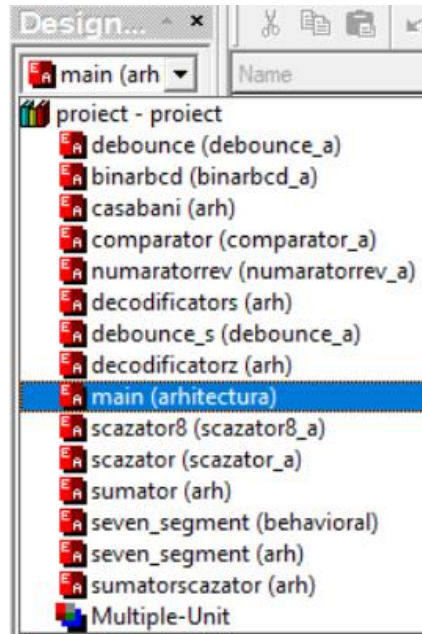
7.2. Active-HDL

In Active-HDL, proiectul se foloseste astfel:

1. Se deschide programul Active-HDL
2. Se compileaza cu ajutorul comenzii „compile all”
3. Se simuleaza cu „New waveform”:



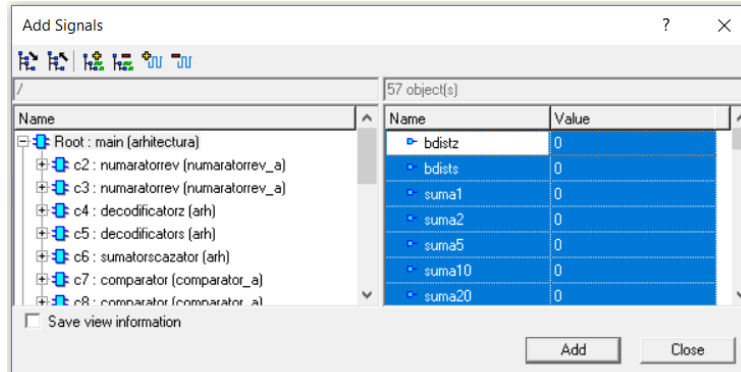
1. Se selecteaza astfel:



2. Se face click dreapta pe fereastra deschisa

3. Se selecteaza „Add signals..”  **Add Signals...** **Ctrl+I**

4. Se adauga toate intrarile, iesirile si eventual, semnalele intermediare, facand click dreapta-> „Select all” si „Add”, iar pentru selectarea semnalelor ce trebuie adaugate, se selecteaza semnalul si se da click pe „Add”




5. Se dau valori intrarilor, facand click dreapta pe lista si selectand „Stimulators”

- se alege clock pentru CLK

-se alege Hotkey pentru restul

6. Se apasa butonul „Run for” 

7. Se apasa pe tastele selectate pentru schimbarea valorilor semnalelor de intrare, apoi se apasa, la fiecare pas, „Run for” 

8. Se urmaresc rezultatele obtinute

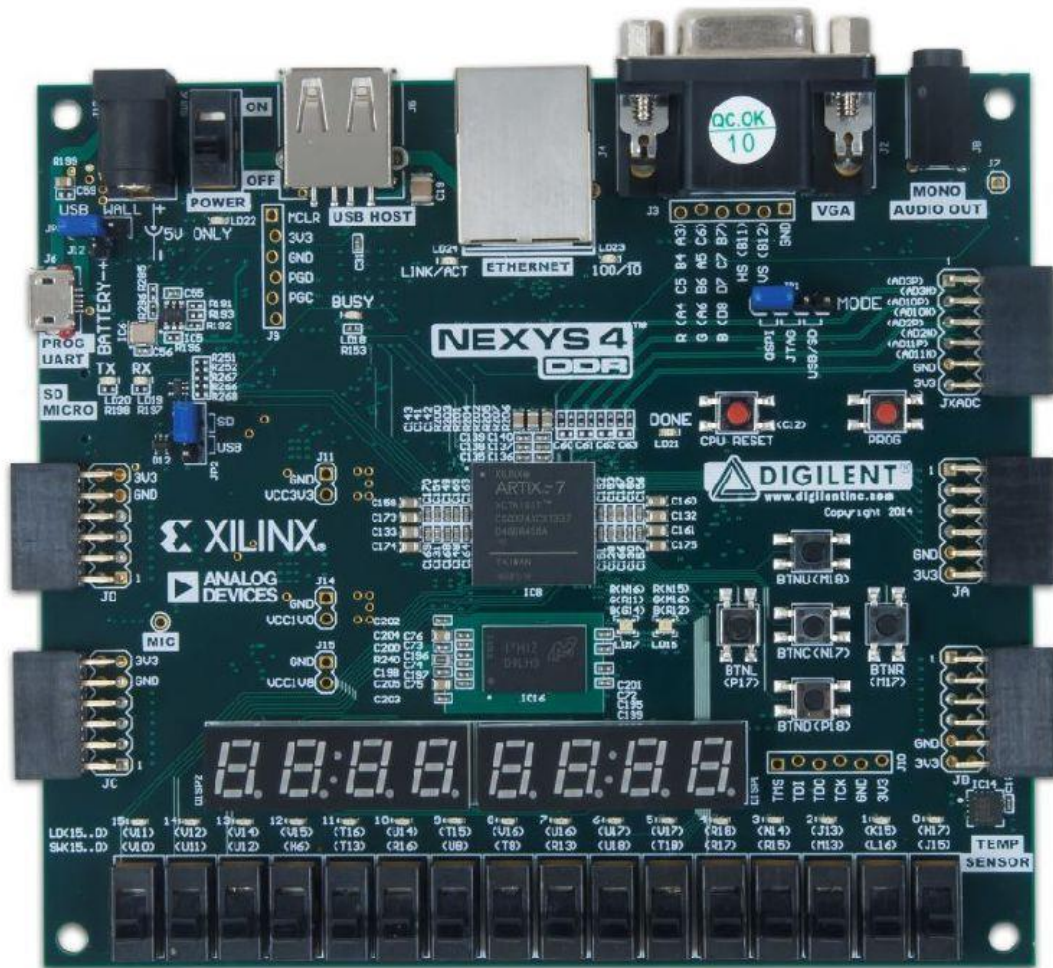
7.3.Xilinx si implementare pe FPGA

Proiectul este compatibil doar cu o placuta FPGA Nexys 4, deoarece sunt utilizate doua blocuri afisoare(8 afisoare 7 segmente), 11 switch-uri, cat si 2 butoane.Pentru afisare se folosesc, pe langa cele 8 afisoare 7 segmente, si 11 leduri.Aceasta se conecteaza printr-un cablu USB la calculator, si se pune sub tensiune, schimbând switch-ul power pe on.

IMPORTANT! La fiecare tranzactie, switch-urile specifice intrarilor enters,enterz si entersuma, se dezactiveaza, iar switch-ul specific intrarii parallelload se activeaza si apoi se dezactiveaza. De aici, o tranzactie se efectueaza dupa cum urmeaza: se apasa pe butoanele specifice bdists,bdistz pentru a se selecta distanta, vizibila si pe afisor, apoi se activeaza switch-urile enters,enterz, zsumaincorecta este aprins. Se introduc bancnotele, comutand switch-urile specifice suma1, suma2, suma5, suma10, suma20, suma50, suma introdusa aparand pe afisor, pe masura ce sunt introduse bancnote.Cand suma introdusa este egala sau depaseste suma ceruta, ledul zsumaincorecta se stinge. Se activeaza entersuma. In acest moment, zbilet se aprinde, iar ledurile specifice bancnotelor date ca rest, se aprind. Daca e imposibil sa se restituie restul, zlipsarest se aprinde, iar daca distanta introdusa este mai mare de 100, zdistantamare se aprinde. Daca nu mai sunt disponibile bilete,zlipsabilete se aprinde. Daca se doreste renuntarea la tranzactie, se comuta switch-ul corespunzator intrarii renunt.

ATENTIE! Este posibil ca, in urma comutarii unui switch, daca acesta nu este utilizat cu foarte mare atentie, sa se interpreteze ca fiind de fapt, doua comutari ale switch-ului.



O placuta FPGA Nexys 4 este prezentata in figura:



In Xilinx, proiectul se foloseste astfel:

1.Se deschide programul Xilinx ISE;

2.Se creeaza un nou spatiu de lucru;

3.Se face click pe "User constraints"  User Constraints , iar apoi se selecteaza "Edit constraints"  Edit Constraints (Text) , unde se declara legaturile;

Exemplu :

```
NET "bdists" LOC=M18 | IOSTANDARD=LVC MOS33 ;
NET "bdistz" LOC=N17 | IOSTANDARD=LVC MOS33 ;
NET "suma1" LOC=V10 | IOSTANDARD=LVC MOS33 ;
NET "suma2" LOC=U11 | IOSTANDARD=LVC MOS33 ;
NET "suma5" LOC=U12 | IOSTANDARD=LVC MOS33 ;
NET "suma10" LOC=H6 | IOSTANDARD=LVC MOS33 ;
NET "suma20" LOC=T13 | IOSTANDARD=LVC MOS33 ;
NET "suma50" LOC=R16 | IOSTANDARD=LVC MOS33 ;
NET "enterz" LOC=T8 | IOSTANDARD=LVC MOS33 ;
NET "enters" LOC=R13 | IOSTANDARD=LVC MOS33 ;
NET "entersuma" LOC=U18 | IOSTANDARD=LVC MOS33 ;
NET "renunt" LOC=L16 | IOSTANDARD=LVC MOS33 ;
NET "CLK" LOC=E3 | IOSTANDARD=LVC MOS33 ;
NET "anod(0)" LOC=J17 | IOSTANDARD=LVC MOS33;
NET "anod(1)" LOC=J18 | IOSTANDARD=LVC MOS33;
NET "anod(2)" LOC=T9 | IOSTANDARD=LVC MOS33;
NET "anod(3)" LOC=J14 | IOSTANDARD=LVC MOS33;
NET "anod(4)" LOC=P14 | IOSTANDARD=LVC MOS33;
NET "anod(5)" LOC=T14 | IOSTANDARD=LVC MOS33;
NET "anod(6)" LOC=K2 | IOSTANDARD=LVC MOS33;
NET "anod(7)" LOC=U13 | IOSTANDARD=LVC MOS33;
NET "catod(6)" LOC=T10 | IOSTANDARD=LVC MOS33;
NET "catod(5)" LOC=R10 | IOSTANDARD=LVC MOS33;
NET "catod(4)" LOC=K16 | IOSTANDARD=LVC MOS33;
NET "catod(3)" LOC=K13 | IOSTANDARD=LVC MOS33;
NET "catod(2)" LOC=P15 | IOSTANDARD=LVC MOS33;
NET "catod(1)" LOC=T11 | IOSTANDARD=LVC MOS33;
NET "catod(0)" LOC=L18 | IOSTANDARD=LVC MOS33;
NET "zbilet" LOC=R18 | IOSTANDARD=LVC MOS33;
NET "z1" LOC=V11 | IOSTANDARD=LVC MOS33;
NET "z2" LOC=V12 | IOSTANDARD=LVC MOS33;
NET "z5" LOC=V14 | IOSTANDARD=LVC MOS33;
NET "z10" LOC=V15 | IOSTANDARD=LVC MOS33;
NET "z20" LOC=T16 | IOSTANDARD=LVC MOS33;
NET "z50" LOC=U14 | IOSTANDARD=LVC MOS33;
NET "zlipsabilete" LOC=N14 | IOSTANDARD=LVC MOS33;
NET "zsumaincorecta" LOC=J13 | IOSTANDARD=LVC MOS33;
NET "zlipsarest" LOC=K15 | IOSTANDARD=LVC MOS33;
```


NET "zdistantamare" LOC=H17 | IOSTANDARD=LVC MOS33;

NET "parallelload" LOC=J15 | IOSTANDARD=LVC MOS33;

4. Se da click pe "Generate Programming File"

5. Se deschide apoi programul „Adept”, folosit pentru a implementa proiectul pe FPGA

6. Se da click pe „Initialize Chain”

7. Se cauta programul facand click pe „Browse” si apoi cautandu-se fisierul cu extensia **.bit**

8. Se da click pe „Program” si urmareste evolutia pe FPGA (se porneste placuta de la switch-ul de power)

8. Posibilitati de dezvoltare ulterioara:

Consider ca nu exista schimbari in ceea ce priveste automatul ,ci doar imbunatatiri. Una dintre ele ar fi introducerea unei comenzi vocale, folositoare persoanelor cu dizabilitati. Alt mod de dezvoltare, ar fi, posibilitatea cererii mai multor bilete in acelasi timp, asa scurtandu-se perioada de asteptare. Totodata, s-ar putea afisa destinatiile posibile, distanta pana la acestea, timpul estimat calatoriei, cat si data si ora plecarii. Utilizatorul ar putea ca, in loc sa introduca distanta, sa selecteze destinatia dorita, sau chiar, suma de care dispune, iar in functie de aceasta, automatul sa ii afiseze destinatiile posibile. De altfel, tranzactiile pentru cumpararea biletelor s-ar putea realiza si cu cardul, introducandu-l pe el in bancomat, urmat de pinul corespunzator.