--Automat MEALY (adresare pe arc)

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_unsigned.all;

entity registru is

port(data\_in:in std\_logic\_vector(3 downto 0);

clk:in bit;

iesire:out std\_logic\_vector(1 downto 0);

sel:out std\_logic\_vector (1 downto 0));

end registru;

architecture arhitectura\_registru of registru is

subtype registrul is std\_logic\_vector(3 downto 0);

signal reg:registrul;

begin

process(clk)

begin

if clk'event and clk='1' then reg<=data\_in;

iesire<=data\_in(1 downto 0);

sel<=data\_in(3 downto 2);

end if;

end process;

end arhitectura\_registru;

entity matrice is

port(data\_mat:in std\_logic\_vector(3 downto 0);

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

end matrice;

architecture arhitectura\_matrice of matrice is

type matrice is array (0 to 15) of std\_logic\_vector(3 downto 0);

begin

process(data\_mat)

variable m:matrice;

begin

m(0):="0000";m(1):="0001";m(2):="0010";

m(3):="0011";m(4):="0100";m(5):="0101";

m(6):="0110";m(7):="0111";m(8):="1000";

m(9):="1001";m(10):="1010";m(11):="1011";

m(12):="1100";m(13):="1101";m(14):="1110";

m(15):="1111";

data\_out<=m(conv\_integer(data\_mat));

end process;

end arhitectura\_matrice;

entity Mealy is

port(intrari:in std\_logic\_vector(1 downto 0);

iesire:out std\_logic\_vector(1 downto 0);

clk:in bit);

end Mealy;

architecture arhitectura\_Mealy of Mealy is

component registru

port(data\_in:in std\_logic\_vector (3 downto 0);

clk:in bit;

iesire:out std\_logic\_vector (1 downto 0);

sel:out std\_logic\_vector (1 downto 0));

end component registru;

component matrice

port(data\_mat:in std\_logic\_vector (3 downto 0);

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

end component matrice;

signal data\_mat:std\_logic\_vector (3 downto 0);

signal data\_out:std\_logic\_vector (3 downto 0);

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

begin

data\_mat<=intrari&sel;

U1:matrice port map(data\_mat,data\_out);

U2:registru port map(data\_out,clk,iesire,sel);

end arhitectura\_Mealy;

--Automat MOORE (adresare pereche)

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_unsigned.all;

entity registru1 is

port(data\_in\_reg1:in std\_logic\_vector (7 downto 0);

sel:out std\_logic\_vector (1 downto 0);

Sua:out std\_logic\_vector (1 downto 0);

Suf:out std\_logic\_vector (1 downto 0);

iesire\_reg1:out std\_logic\_vector(1 downto 0));

end registru1;

architecture arhitectura\_registru of registru1 is

begin

process(data\_in\_reg1)

begin

sel<=data\_in\_reg1(7 downto 6);

Sua<=data\_in\_reg1(5 downto 4);

Suf<=data\_in\_reg1(3 downto 2);

iesire\_reg1<=data\_in\_reg1(1 downto 0);

end process;

end arhitectura\_registru;

entity registru2 is

port(data\_in\_reg2:in std\_logic\_vector (1 downto 0);

clk:bit;

iesire\_reg2:out std\_logic\_vector (1 downto 0));

end registru2;

architecture arhitectura\_registru2 of registru2 is

begin

process(clk)

begin

if clk'event and clk='1' then iesire\_reg2<=data\_in\_reg2;

end if;

end process;

end arhitectura\_registru2;

entity matrice is

port(data\_mat:in std\_logic\_vector(3 downto 0);

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

end matrice;

architecture arhitectura\_matrice of matrice is

type matrice is array (0 to 15) of std\_logic\_vector(3 downto 0);

begin

process(data\_mat)

variable m:matrice;

begin

m(0):="0000";m(1):="0001";m(2):="0010";

m(3):="0011";m(4):="0100";m(5):="0101";

m(6):="0110";m(7):="0111";m(8):="1000";

m(9):="1001";m(10):="1010";m(11):="1011";

m(12):="1100";m(13):="1101";m(14):="1110";

m(15):="1111";

data\_out<=m(conv\_integer(data\_mat));

end process;

end arhitectura\_matrice;

entity mux1 is

port(A,B:in std\_logic\_vector (1 downto 0);

selectie:std\_logic;

C:out std\_logic\_vector (1 downto 0));

end mux1;

architecture arhitectura\_mux1 of mux1 is

begin

C<=A when selectie='1' else B;

end arhitectura\_mux1;

entity Mux2 is

port(intrari:in std\_logic\_vector (3 downto 0);

selectie\_mux2:in std\_logic\_vector (1 downto 0);

rezultat\_mux2:out std\_logic);

end Mux2;

architecture arhitectura\_Mux2 of Mux2 is

begin

process(intrari)

begin

case selectie\_mux2 is

when "00"=>rezultat\_mux2<=intrari(0);

when "01"=>rezultat\_mux2<=intrari(1);

when "10"=>rezultat\_mux2<=intrari(2);

when "11"=>rezultat\_mux2<=intrari(3);

when others=>

end case;

end process;

end arhitectura\_Mux2;

entity Moore is

port(intrari:in std\_logic\_vector (3 downto 0);

clk:in bit;

iesire:out std\_logic\_vector (1 downto 0));

end Moore;

architecture Arhitectura\_Moore of Moore is

component registru1

port(data\_in\_reg1:in std\_logic\_vector (7 downto 0);

sel:out std\_logic\_vector (1 downto 0);

Sua:out std\_logic\_vector (1 downto 0);

Suf:out std\_logic\_vector (1 downto 0);

iesire\_reg1:out std\_logic\_vector(1 downto 0));

end component registru1;

component registru2

port(data\_in\_reg2:in std\_logic\_vector (1 downto 0);

clk:bit;

iesire\_reg2:out std\_logic\_vector (1 downto 0));

end component registru2;

component matrice

port(data\_in\_mat:in std\_logic\_vector(1 downto 0);

data\_out\_mat:out std\_logic\_vector (7 downto 0));

end component matrice;

component Mux2

port(intrari:in std\_logic\_vector (3 downto 0);

selectie\_mux2:in std\_logic\_vector (1 downto 0);

rezultat\_mux2:out std\_logic);

end component mux2;

component mux1

port(A,B:in std\_logic\_vector (1 downto 0);

selectie:std\_logic;

C:out std\_logic\_vector (1 downto 0));

end component mux1;

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

signal rezultat\_mux2:std\_logic;

signal Sua,Suf:std\_logic\_vector (1 downto 0);

signal C,iesire\_reg2:std\_logic\_vector (1 downto 0);

signal data\_out\_mat:std\_logic\_vector (7 downto 0);

begin

U1:mux2 port map (intrari,sel,rezultat\_mux2);

U2:mux1 port map (Sua,Suf,rezultat\_mux2,C);

U3:registru2 port map(C,clk,iesire\_reg2);

U4:matrice port map (iesire\_reg2,data\_out\_mat);

U5:registru1 port map (data\_out\_mat,sel,Sua,Suf,iesire);

end Arhitectura\_Moore;

-- Automat\_adresare pe stare cu adresa PRESUPUSA

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity MUX is

port(input: in std\_logic\_vector(7 downto 0);

sel:in std\_logic\_vector(2 downto 0);

z:out std\_logic);

end ;

architecture ARH\_MUX of MUX is

begin

process(input,sel)

begin

case sel is

when "000" => z<=input(0);

when "001" => z<=input(1);

when "010" => z<=input(2);

when "011" => z<=input(3);

when "100" => z<=input(4);

when "101" => z<=input(5);

when "110" => z<=input(6);

when "111" => z<=input(7);

when others => z<='0';

end case;

end process;

end architecture;

entity MEM is

port(adr:inout std\_logic\_vector(3 downto 0);

iesire: out std\_logic\_vector(10 downto 0):="00000000000");

end;

architecture ARH\_MEM of MEM is

type matrice is array (0 to 15) of std\_logic\_vector(10 downto 0);

begin

process(adr)

variable m:matrice;

begin

m(0):="00000000000";m(1):="00000000001";

m(2):="00000000010";m(3):="00000000011";

m(4):="00000000100";m(5):="00000000101";

m(6):="00000000110";m(7):="00000000111";

m(8):="00000001000";m(9):="00000001001";

m(10):="00000001010";m(11):="00000001011";

m(12):="00000001100";m(13):="00000001101";

m(14):="00000001110";m(15):="00000001111";

iesire<=m(conv\_integer(adr));

end process;

end architecture;

entity Registru is

port(x: in std\_logic\_vector(10 downto 0);

Sel:out std\_logic\_vector(2 downto 0);

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

iesire:out std\_logic\_vector(3 downto 0):="0000");

end entity;

architecture ARH\_REG of Registru is

begin

Sel<=x(10 downto 8);

Sua<=x(7 downto 4);

iesire<=x(3 downto 0);

end architecture;

entity Logica\_comb is

port(A:in std\_logic\_vector(3 downto 0);

B:in std\_logic;

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

INC:out std\_logic);

end entity;

architecture ARH\_LOG of Logica\_comb is

begin

process(A,B)

begin

if(B='0') then INC<='1';

else INC<='1';

end if;

Sua<=A;

end process;

end architecture;

entity numarator is

port(CLK:in bit;

INC: in std\_logic;

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

adr:inout std\_logic\_vector(3 downto 0):="0000");

end entity;

architecture ARH\_Num of numarator is

begin

process(CLK,INC)

begin

if(CLK'event and CLK='1') then

if(INC='1') then adr<=adr+"0001";

else adr<=Sua;

end if;

end if;

end process;

end architecture;

entity Automat is

port(CLK:in bit;

intrari:in std\_logic\_vector(7 downto 0);

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

end entity;

architecture ARH\_AUT of Automat is

component MEM is --asta este memoria automatului port(adr:inout std\_logic\_vector(3 downto 0);

iesire: out std\_logic\_vector(10 downto 0):="00000000000");

end component;

component Registru is

port(x: in std\_logic\_vector(10 downto 0);

Sel:out std\_logic\_vector(2 downto 0);

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

iesire:out std\_logic\_vector(3 downto 0):="0000");

end component;

component Logica\_comb is

port(A:in std\_logic\_vector(3 downto 0);

B:in std\_logic;

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

INC:out std\_logic:='0');

end component;

component MUX is

port(input: in std\_logic\_vector(7 downto 0);

sel:in std\_logic\_vector(2 downto 0);

z:out std\_logic);

end component;

component numarator is

port(CLK:in bit;

INC: in std\_logic;

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

adr:inout std\_logic\_vector(3 downto 0):="0000");

end component;

signal adr: std\_logic\_vector(3 downto 0):="0000";

signal ROM\_out:std\_logic\_vector(10 downto 0);

signal sel:std\_logic\_vector(2 downto 0);

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

signal mux\_out:std\_logic;

signal INC:std\_logic;

begin

P1: MEM port map (adr,ROM\_out);

P2: registru port map(ROM\_out,sel,Sua,z);

P3: MUX port map(intrari,sel,mux\_out);

P4: Logica\_comb port map(Sua,mux\_out,Sua,INC);

P5: numarator port map(CLK,INC,Sua,adr);

end architecture;

-- Automat\_adresare pe stare cu adresa VARIABILA

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

use IEEE.STD\_LOGIC\_UNSIGNED.all;

entity MUX is

port(input: in std\_logic\_vector(7 downto 0);

sel:in std\_logic\_vector(2 downto 0);

z:out std\_logic);

end ;

architecture ARH\_MUX of MUX is

begin

process(input,sel)

begin

case sel is

when "000" => z<=input(0);

when "001" => z<=input(1);

when "010" => z<=input(2);

when "011" => z<=input(3);

when "100" => z<=input(4);

when "101" => z<=input(5);

when "110" => z<=input(6);

when "111" => z<=input(7);

when others => null;

end case;

end process;

end architecture;

entity MEM is

port(adr: inout std\_logic\_vector(3 downto 0);

iesire: out std\_logic\_vector(10 downto 0));

end;

architecture ARH\_MEM of MEM is

type matrice is array (0 to 15) of std\_logic\_vector(10 downto 0);

begin

process(adr)

variable m:matrice;

begin

m(0):="00000000000";m(1):="00000000001";

m(2):="00000000010";m(3):="00000000011";

m(4):="00000000100";m(5):="00000000101";

m(6):="00000000110";m(7):="00000000111";

m(8):="00000001000";m(9):="00000001001";

m(10):="00000001010";m(11):="00000001011";

m(12):="00000001100";m(13):="00000001101";

m(14):="00000001110";m(15):="00000001111";

iesire<=m(conv\_integer(adr));

end process;

end architecture;

entity Registru is

port(x: in std\_logic\_vector(10 downto 0);

format:in bit;

Sel:out std\_logic\_vector(2 downto 0);

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

iesire:out std\_logic\_vector(3 downto 0):="0000");

end entity;

architecture ARH\_REG of Registru is

begin

Sel<=x(10 downto 8);

Sua<=x(7 downto 4);

iesire<=x(3 downto 0);

end architecture;

entity Logica\_comb is

port(A:in std\_logic\_vector(3 downto 0);

B:in std\_logic;

format:in bit;

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

INC:out std\_logic:='0');

end entity;

architecture ARH\_LOG of Logica\_comb is

begin

process(A,B)

begin

if(B='0') then INC<='1';

else INC<='1';

end if;

Sua<=A;

end process;

end architecture;

entity numarator is

port(CLK:in bit;

INC: in std\_logic;

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

adr:inout std\_logic\_vector(3 downto 0):="0000");

end entity;

architecture ARH\_Num of numarator is

begin

process(CLK,INC)

begin

if(CLK'event and CLK='1') then

if(INC='1') then adr<=adr+"0001";

else adr<=Sua;

end if;

end if;

end process;

end architecture;

entity Automat is

port(CLK:in bit;

intrari:in std\_logic\_vector(7 downto 0);

z:out std\_logic\_vector(3 downto 0):="0000");

end entity;

architecture ARH\_AUT of Automat is

component MEM is --asta este memoria automatului port(adr:inout std\_logic\_vector(3 downto 0);

iesire: out std\_logic\_vector(10 downto 0):="00000000000");

end component;

component Registru is

port(x: in std\_logic\_vector(10 downto 0);

format:in bit;

Sel:out std\_logic\_vector(2 downto 0);

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

iesire:out std\_logic\_vector(3 downto 0):="0000");

end component;

component Logica\_comb is

port(A:in std\_logic\_vector(3 downto 0);

B:in std\_logic;

format:in bit;

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

INC:out std\_logic:='0');

end component;

component MUX is

port(input: in std\_logic\_vector(7 downto 0);

sel:in std\_logic\_vector(2 downto 0);

z:out std\_logic);

end component;

component numarator is

port(CLK:in bit;

INC: in std\_logic;

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

adr:inout std\_logic\_vector(3 downto 0):="0000");

end component;

signal adr: std\_logic\_vector(3 downto 0):="0000";

signal ROM\_out:std\_logic\_vector(10 downto 0);

signal sel:std\_logic\_vector(2 downto 0);

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

signal mux\_out:std\_logic;

signal INC:std\_logic;

signal format:bit;

begin

P1: MEM port map (adr,ROM\_out);

P2: registru port map(ROM\_out,format,sel,Sua,z);

P3: MUX port map(intrari,sel,mux\_out);

P4: Logica\_comb port map(Sua,mux\_out,format,Sua,INC);

P5: numarator port map(CLK,INC,Sua,adr);

end architecture;

--Automat LINIAR

-- T = x+ D^2\*x + D^3\*x

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity bistabil\_D is

port(D\_in:in std\_logic;

clk:in bit;

D\_out:out std\_logic);

end bistabil\_D;

architecture arhitectura\_bistabil\_D of bistabil\_D is

begin

process(clk)

begin

if clk'event and clk='1' then D\_out<=D\_in;

end if;

end process;

end arhitectura\_bistabil\_D;

entity sumator is

port( a,b:in std\_logic;

c:out std\_logic);

end sumator;

architecture arhitectura\_sumator of sumator is

begin

c<=a xor b;

end arhitectura\_sumator;

entity Automat\_liniar is

port(clk:in bit;

intrare:in std\_logic;

iesire:out std\_logic);

end Automat\_liniar;

architecture Arhitectura\_Automat\_liniar of Automat\_liniar is

component bistabil\_d

port(D\_in:in std\_logic;

clk:in bit;

D\_out:out std\_logic);

end component bistabil\_d;

component sumator

port(a,b:in std\_logic;

c:out std\_logic);

end component sumator;

signal iesire\_bistabil\_3,iesire\_bistabil\_2,iesire\_bistabil\_1,iesire\_sumator\_1

:std\_logic;

begin

D3:bistabil\_d port map (intrare,clk,iesire\_bistabil\_3);

S1:sumator port map(iesire\_bistabil\_3,intrare,iesire\_sumator\_1);

D2:bistabil\_d port map (iesire\_sumator\_1,clk,iesire\_bistabil\_2);

D1:bistabil\_d port map (iesire\_bistabil\_2,clk,iesire\_bistabil\_1);

S2:sumator port map(iesire\_bistabil\_1,intrare,iesire);

end Arhitectura\_Automat\_liniar;