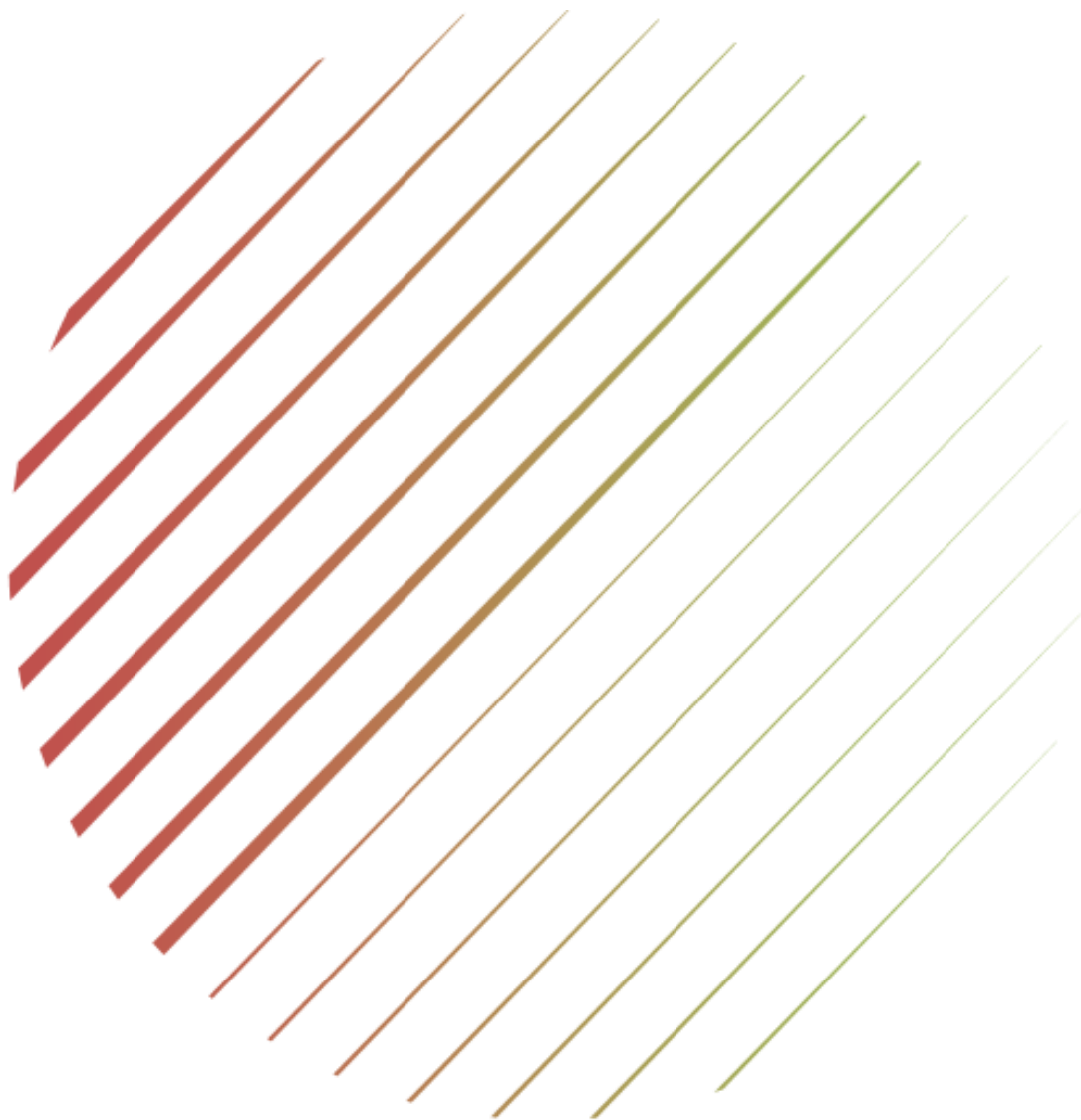




UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA



AUTOMAT BANCAR
31. 05. 2023

Cotei Ruxanda-Maria &
Grigoraş Victor-Andrei
Coordonator: Maier Noema-Laura

Cuprins:

Specificații:	3
Proiectare și implementare:	3
Inițializare:	3
Configurare plăcuță FPGA:	4
Schemă bloc:.....	4
Unitate de comandă:	5
Organigramă:	6
Schema logică detaliată:.....	11
Reprezentarea resurselor:	12
Codul din spatele fiecărei resurse:	15
Instrucțiuni de întreținere și utilizare:	33
Justificarea soluției alese:	35
Posibilități de dezvoltare ulterioare:	36
Bibliografie:	37

Specificatii:

Să se proiecteze un automat bancar pentru extrageri de sume în EURO. Se presupune că suma maximă care poate fi extrasă o dată este de maximum 1.000 euro. Inițial se efectuează identificarea cardului și se alege operația. Vor fi suportate minim 4 carduri/conturi diferite și se vor implementa minimum 4 operații diferite. Automatul dispune de o casă în care inițial se introduce o anumită sumă (număr de bancnote de diferite valori). În cazul cererii de eliberare de numerar se introduce suma, se verifică existența sumei cerute, se vizualizează tipurile de bancnote emise și se actualizează contul. Apoi se eliberează cardul, suma și, eventual, chitanța.

Proiectare și implementare:

Proiectul descrie comportarea unui automat bancar în următoarele patru situații:

- interogare sold;
- retragere numerar;
- depunere numerar;
- schimbare PIN.

Inițializare:

Înainte de începerea propriu-zisă a interacțiunii dintre automat și utilizator, bancomatul este inițializat cu un număr cunoscut de bancnote de fiecare tip, cât și cu cele 4 PIN-uri după care se identifică cele 4 carduri, pentru fiecare din acestea cunoscându-se soldul disponibil pe card.

Operațiunea de interogare sold are ca efect afișarea pe SSD a sumei ce se afla pe card.

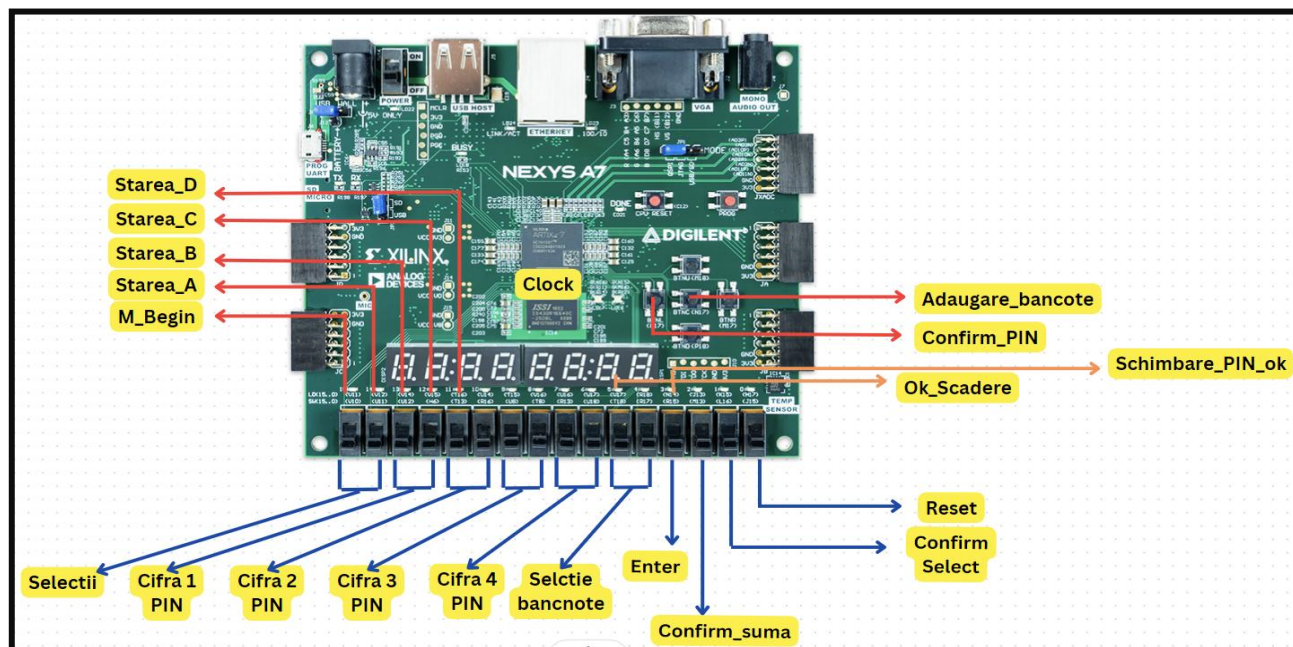
Operațiunea de retragere numerar presupune introducerea unei sume, compararea acesteia cu suma aflată pe cardul clientului, verificarea cu bancnotele din bancomat. Dacă bancomatul are bancnotele necesare, se eliberează (se scad) acestea și se scade suma din cont cu suma introdusă. În caz contrar, se verifică dacă se poate obține suma cerută din alte bancnote, se procedează la fel ca în situația expusă anterior.

Operațiunea de depunere numerar constă în introducerea sumei de către utilizator, urmată de creșterea valorii din cont și a bancnotelor din casa automatului.

Operațiunea de schimbare PIN necesită introducerea unui nou cod PIN, verificarea ca acesta să nu existe deja (nu pot fi două PIN-uri identice) și schimbarea acestuia în memoria bancomatului.

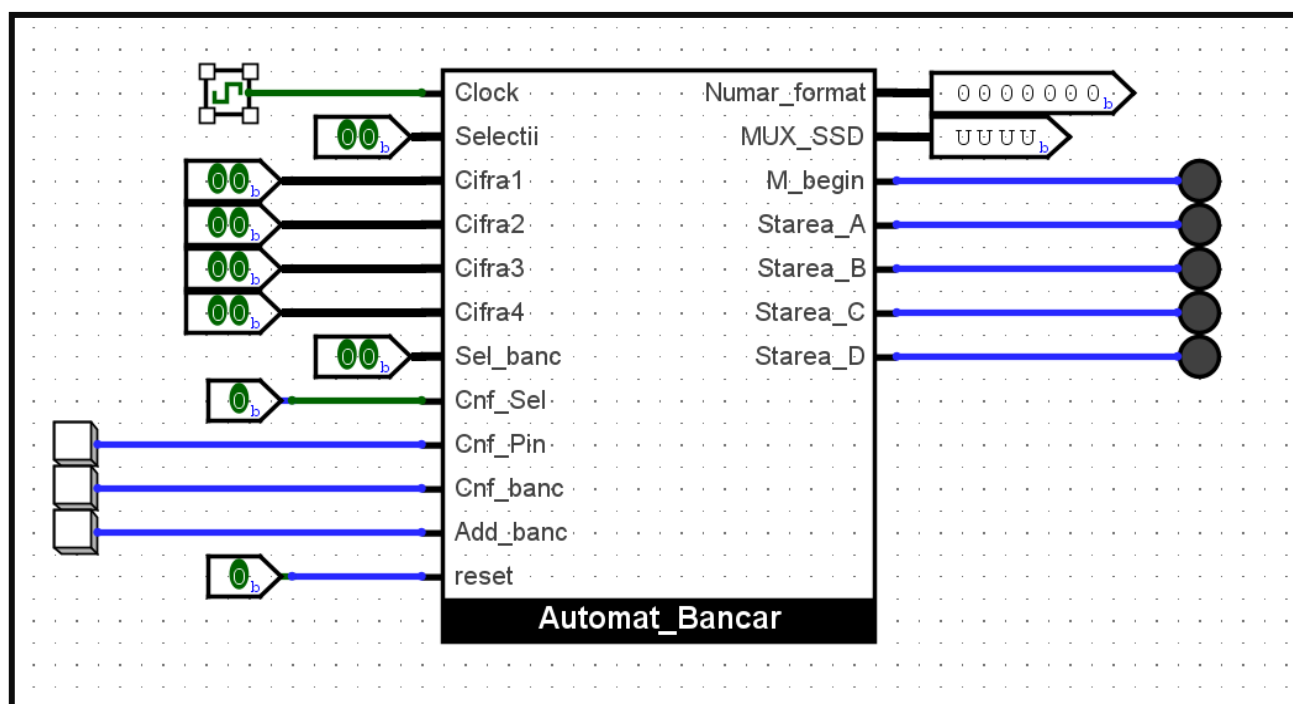
Configurare plăcuță FPGA:

Pentru implementarea proiectului se va folosi ca resursă hardware o plăcuță FPGA NEXYS Artix-7 (A7).



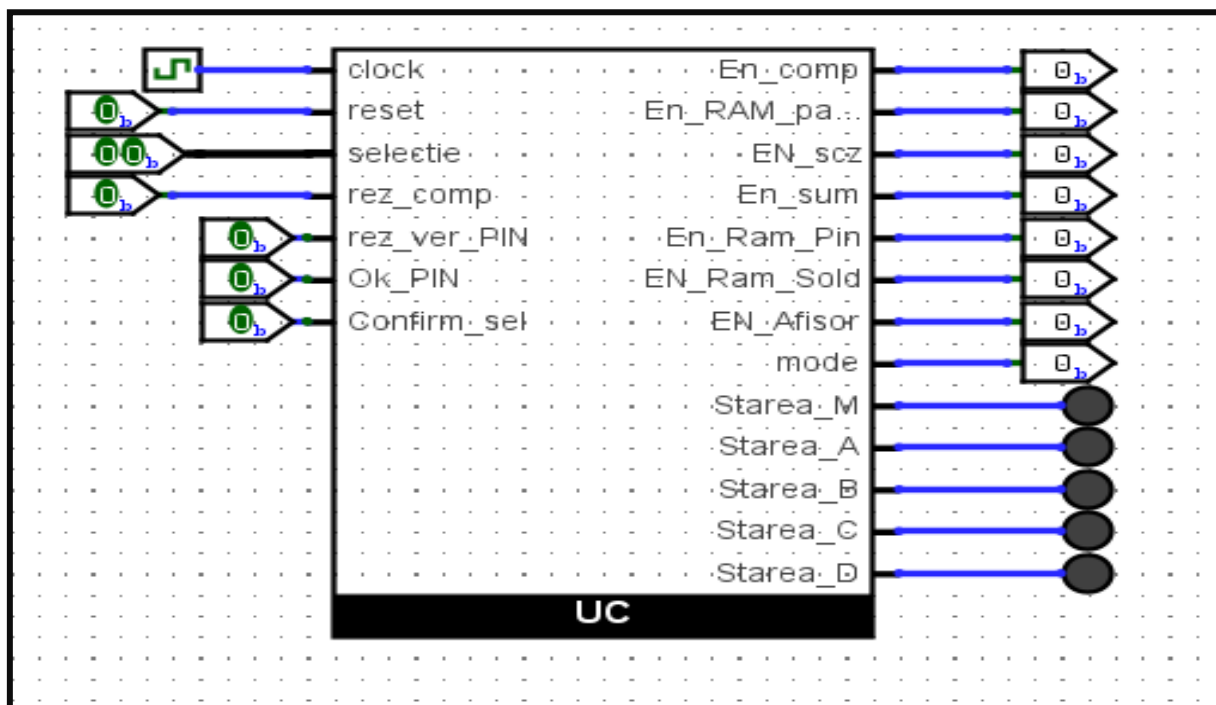
Schemă bloc:

Schema bloc (black-box-ul proiectului) conține toate intrările și ieșirile circuitului, tot ce are legătură cu interacțiunea cu utilizatorul.



Unitate de comandă:

Unitatea de comandă este formată din toate interacțiunile componentelor între ele, asigurând buna funcționare a bancomatului. Aceasta conține atât semnalele ce se transmit între elemente (semnale care nu sunt vizibile utilizatorului), cât și stările prin care trece automatul pentru a efectua operațiile necesare.



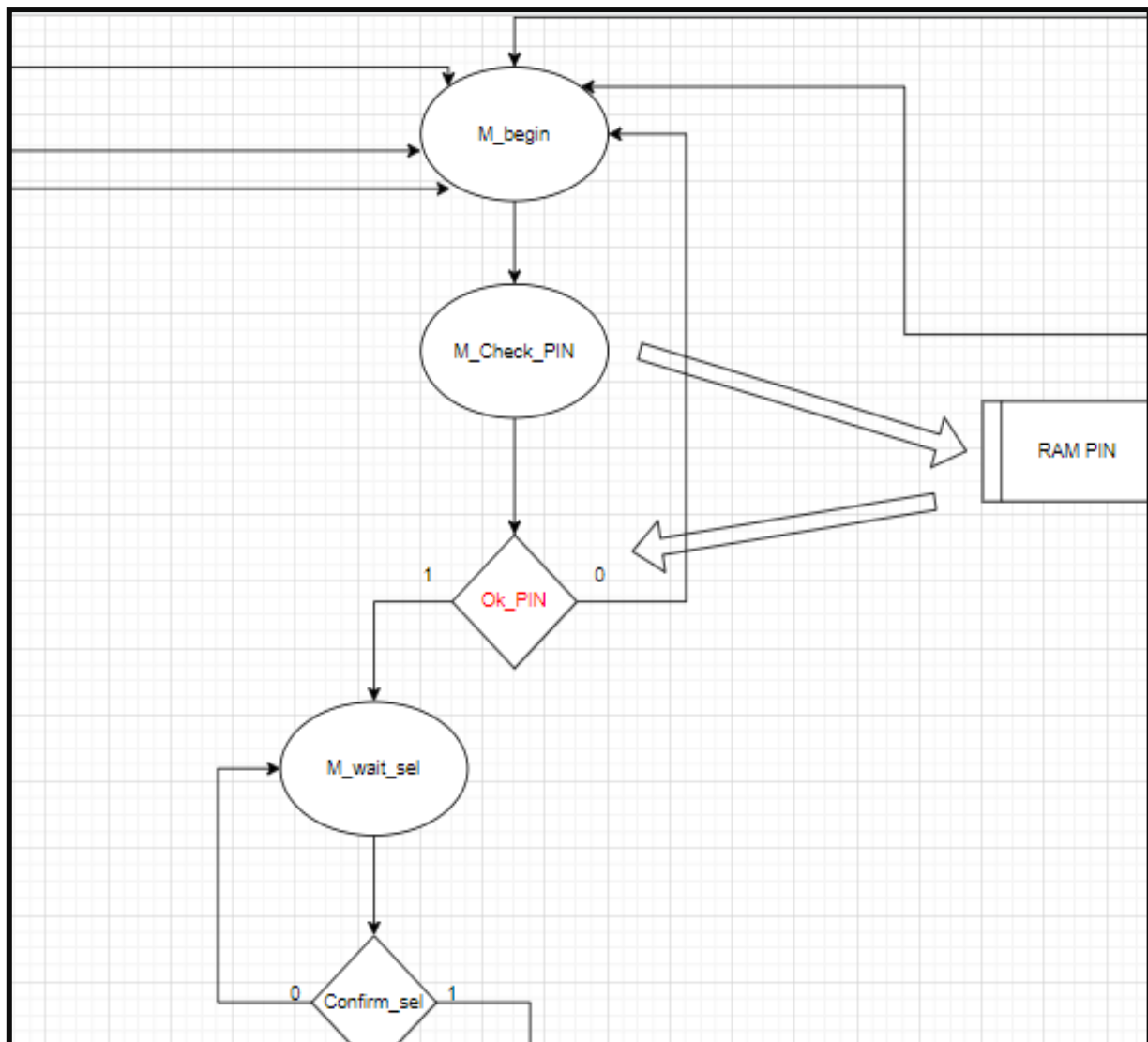
Unitatea de comandă este proiectată pe baza organigramei. Organigrama reprezintă o digramă de stări ce detaliază pas cu pas funcționalitatea automatului, pentru o mai bună înțelegere. Aceasta este alcătuită pe baza a trei componente simple: stare (oval), decizie (romb), ieșire (dreptunghi), la care s-au adăugat elementele din unitatea de execuție (cu săgeți mai groase), pentru o înțelegere mai ușoară.

Organigrama proiectului este structurată în 5 categorii, în funcție de stări și operații, fiind codificate astfel:

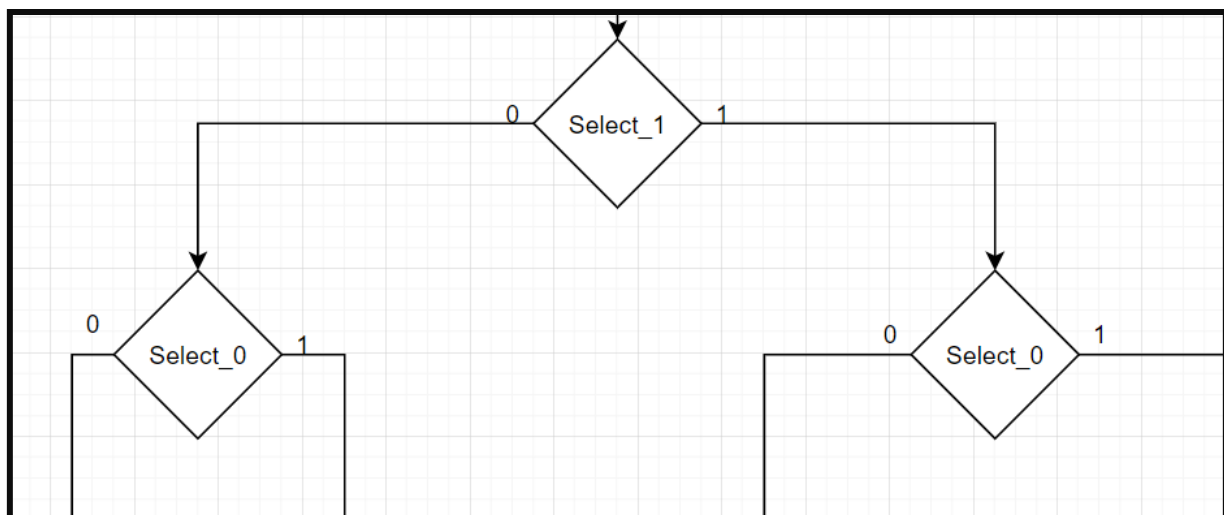
- M – inițializare și selectare operații;
- A – afișare sold;
- B – retragere numerar;
- C – depunere numerar;
- D – schimbare PIN.

Organigramă:

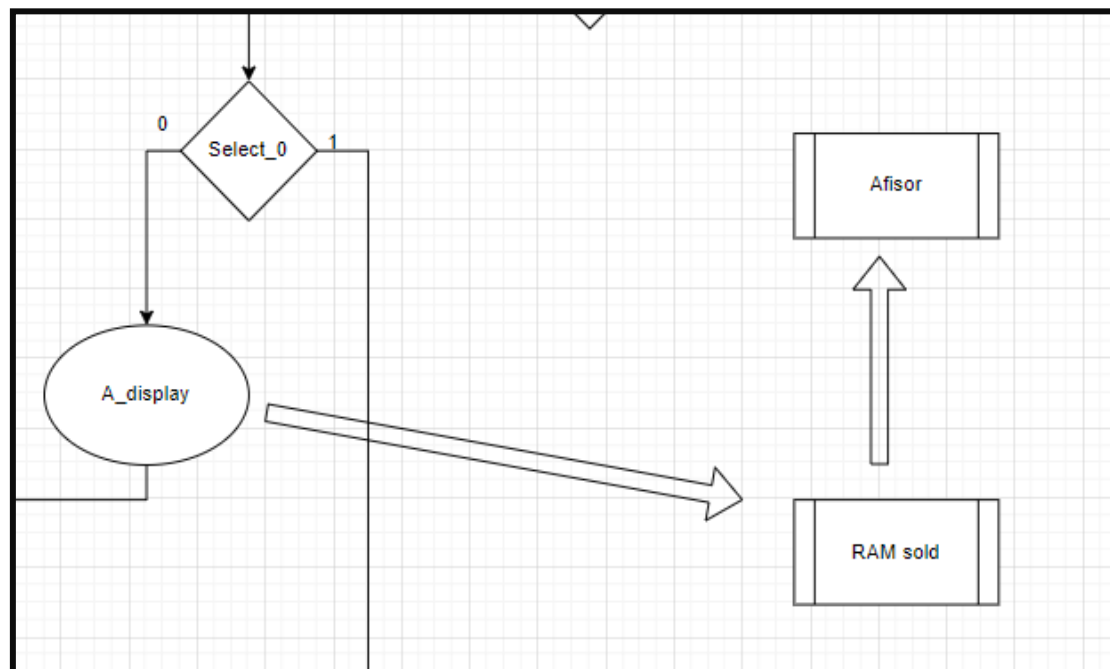
- Stările M:



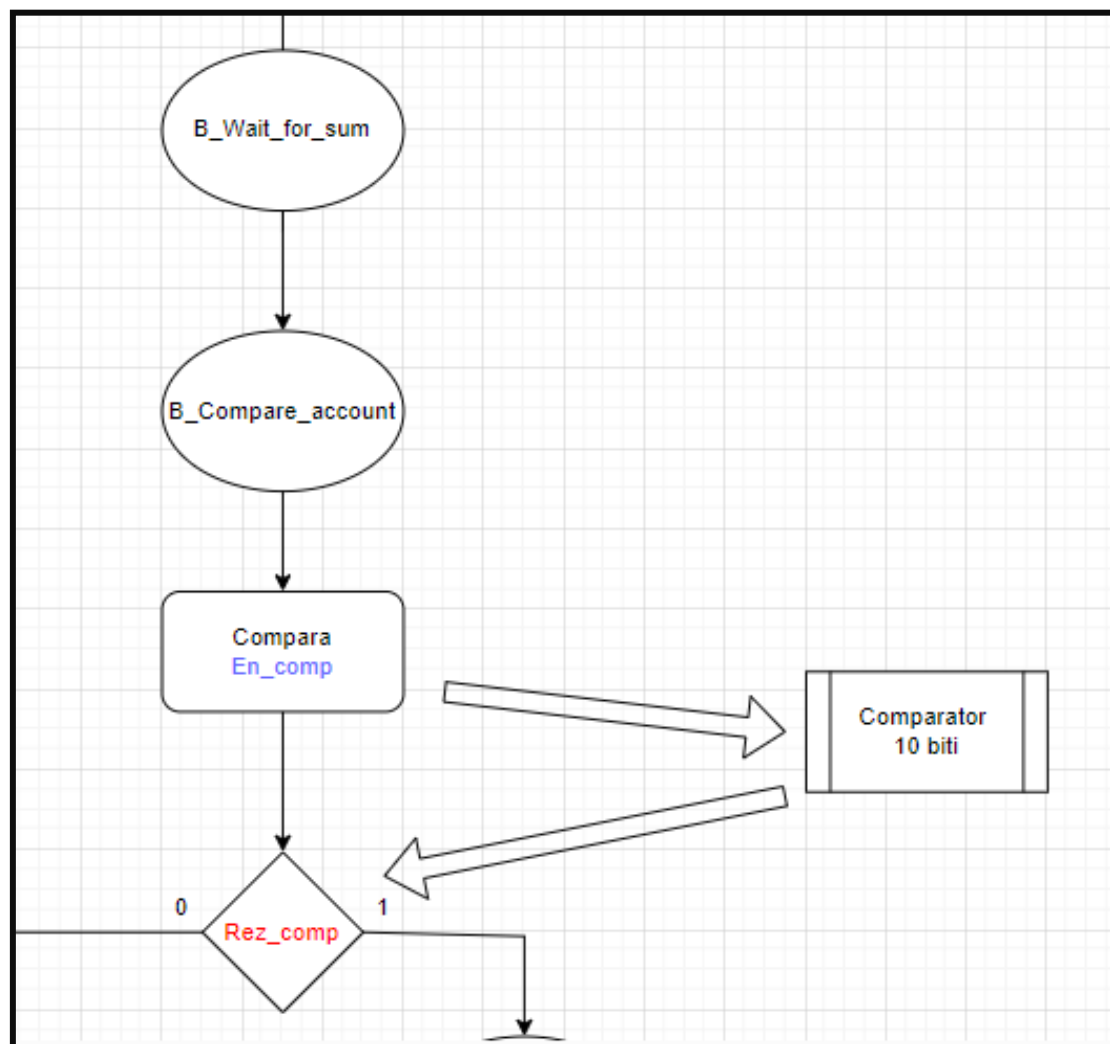
- Selecții:

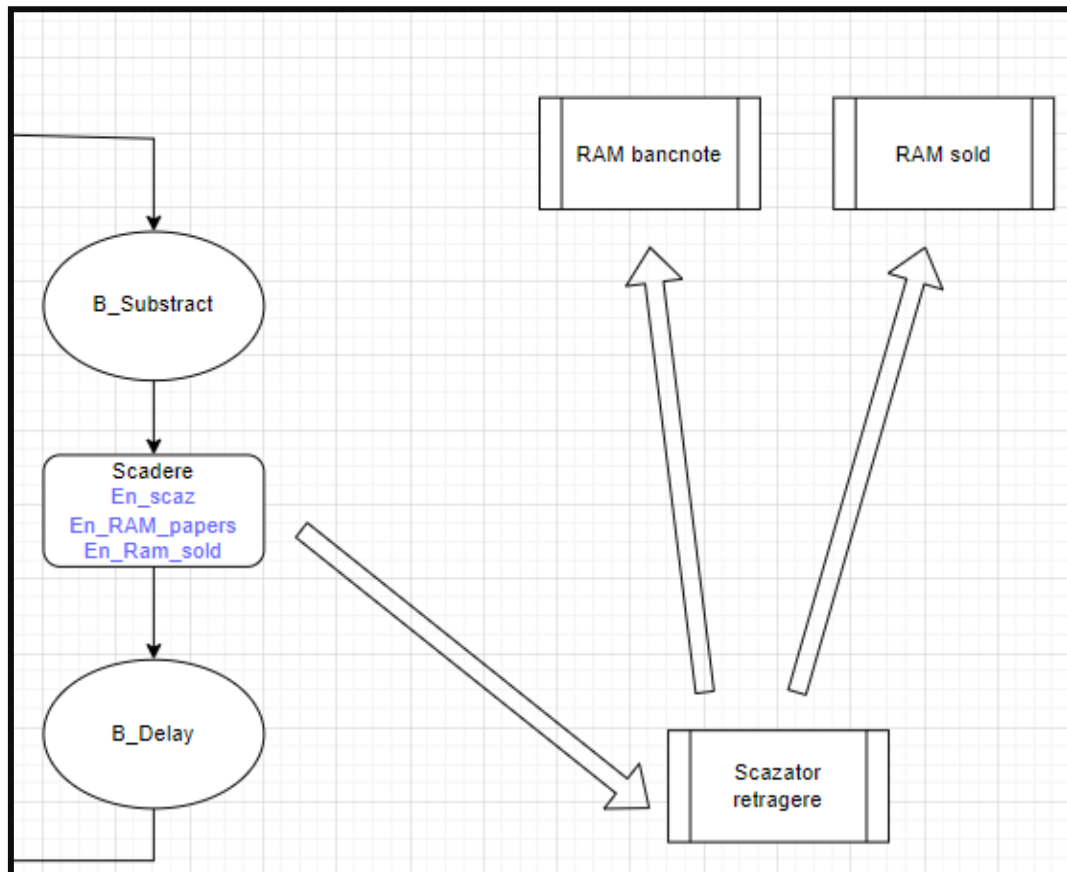


- Starea A:

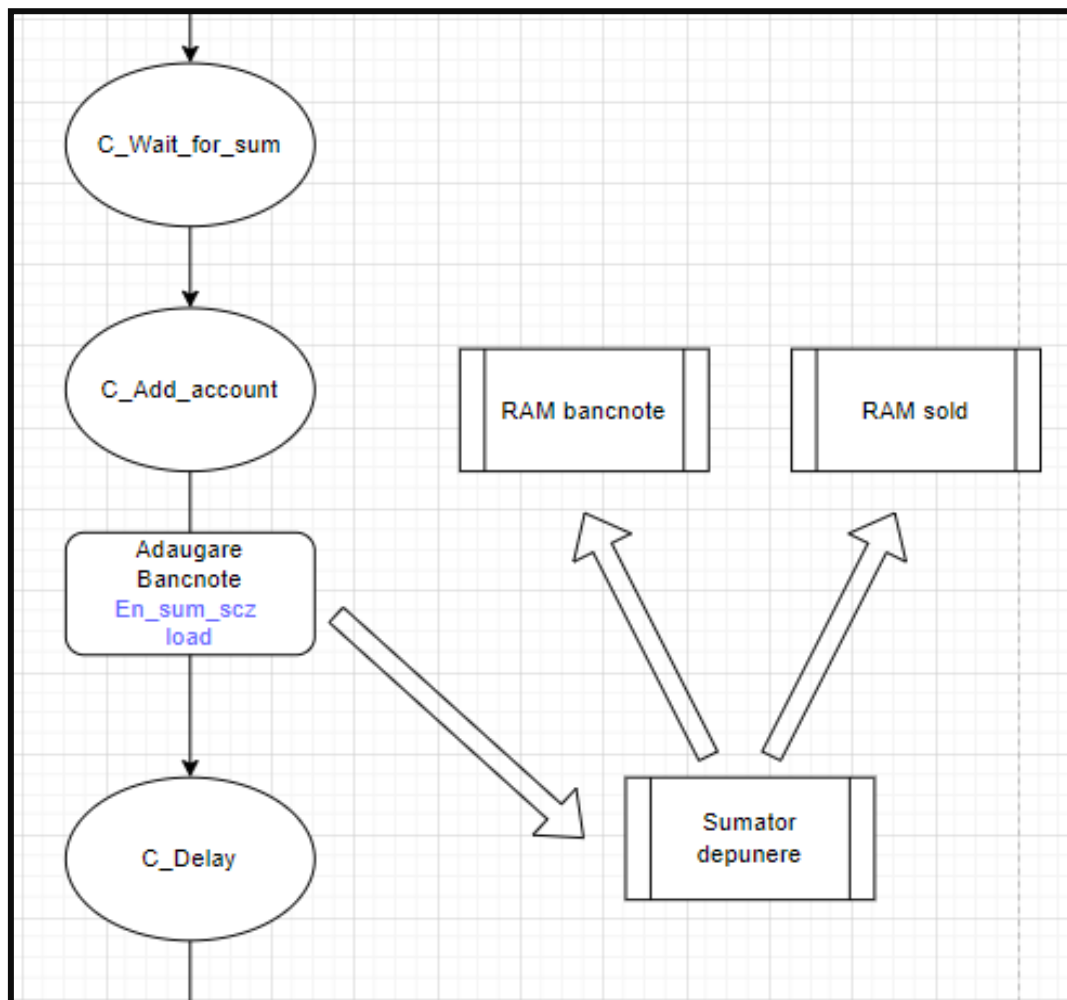


- Stările B:

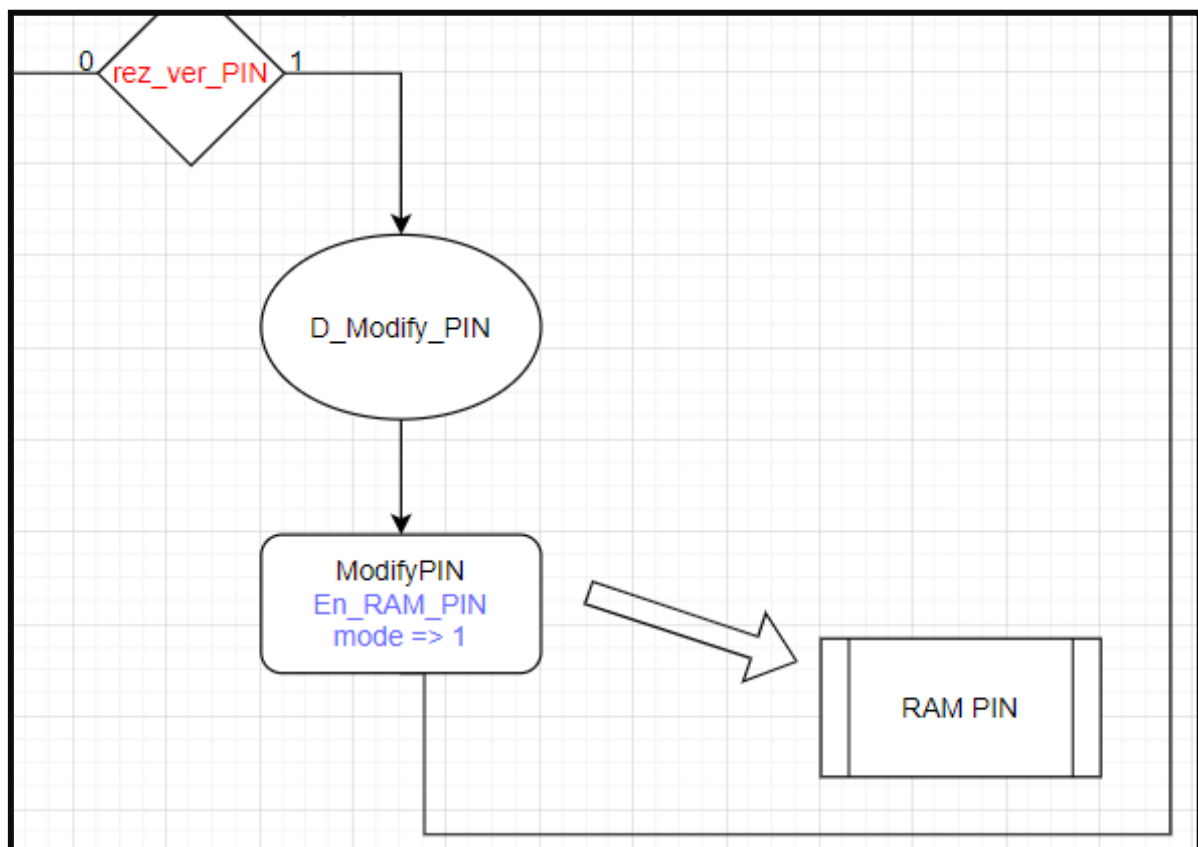
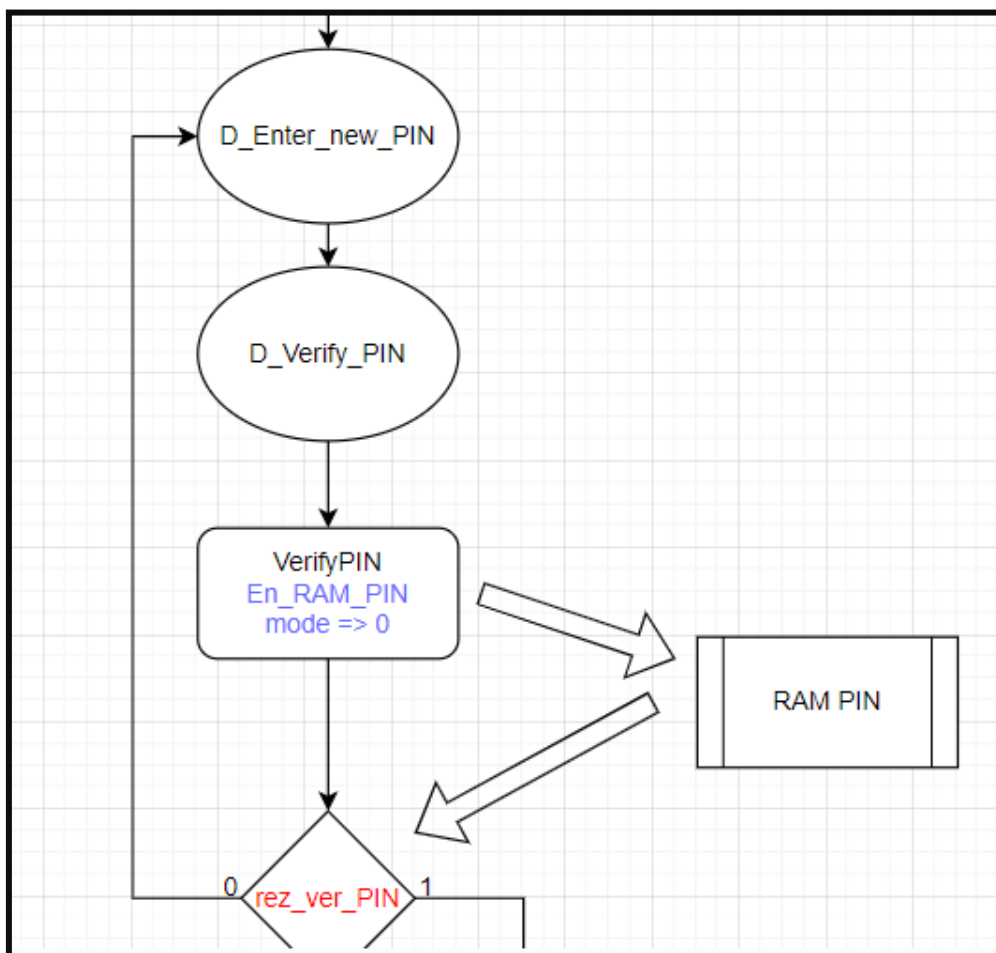


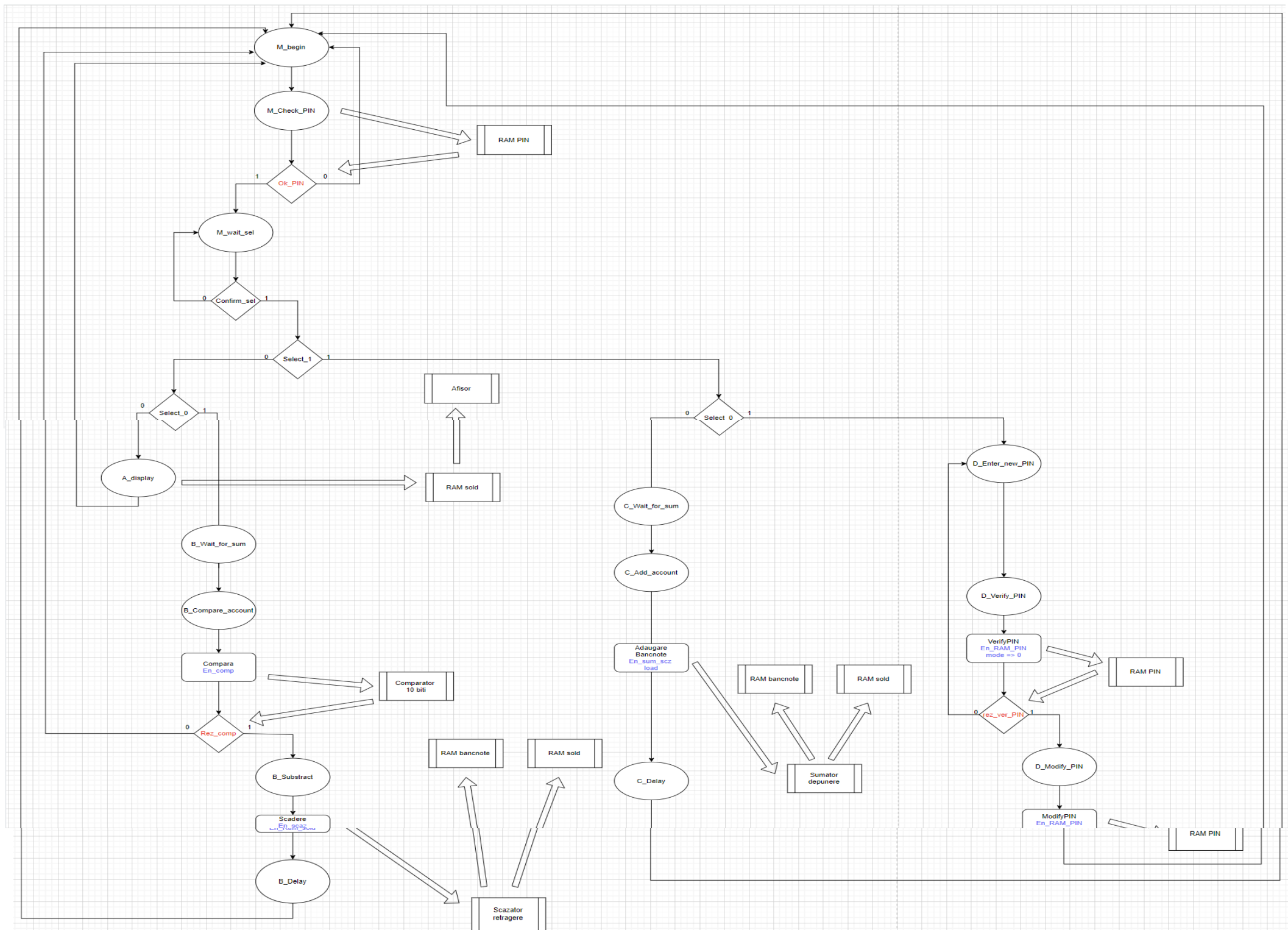


- Stările C:

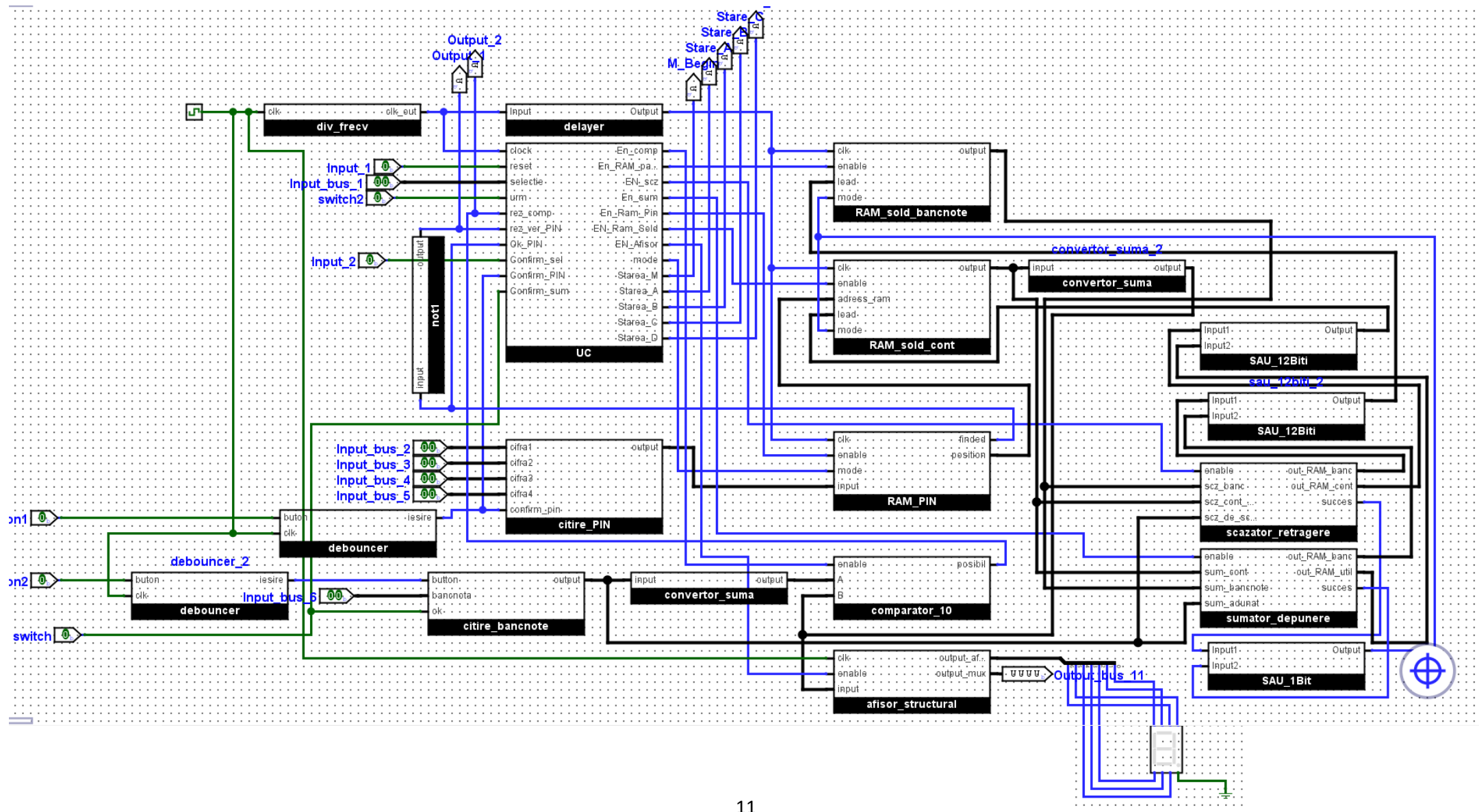


- Stările D:


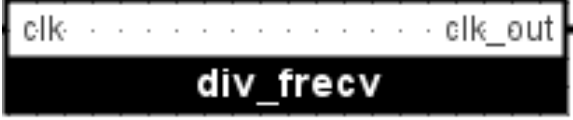
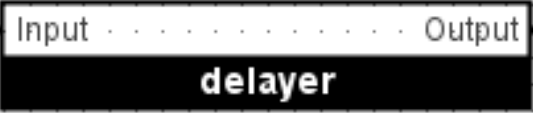

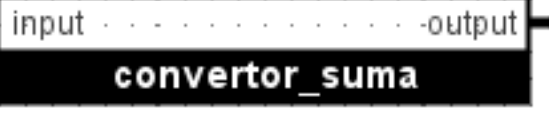



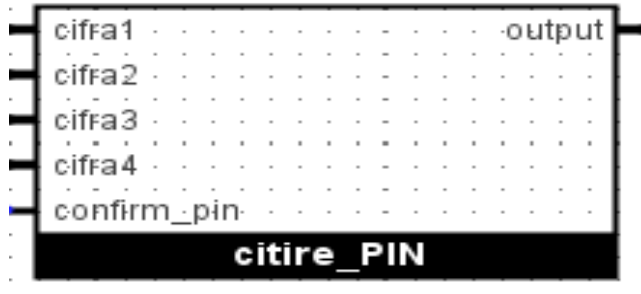
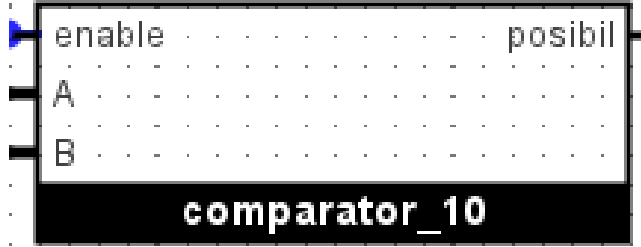
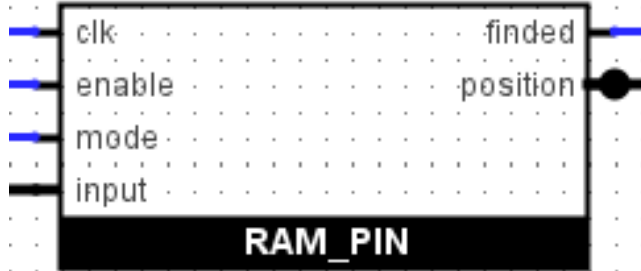

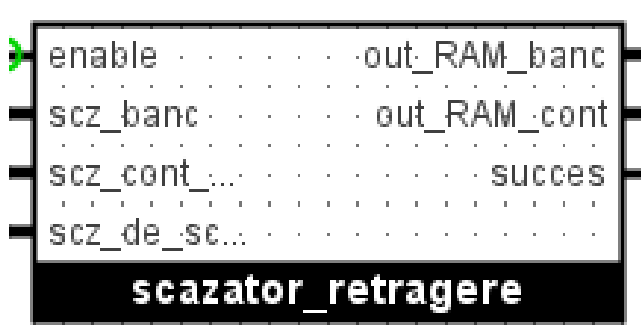


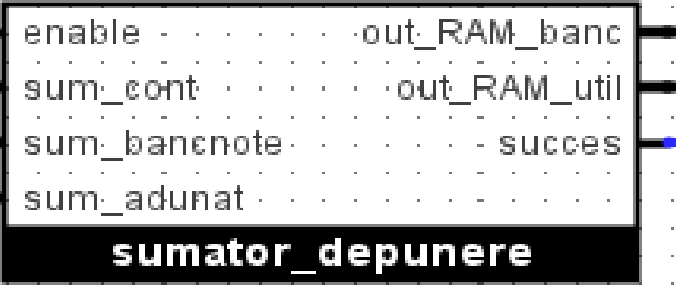
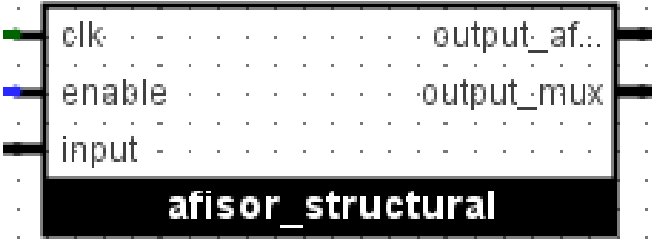
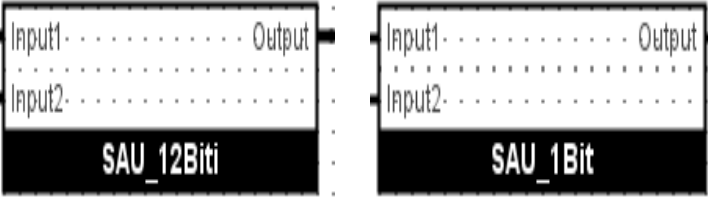

Schema logică detaliată:



Reprezentarea resurselor:

	<p>Debouncerul are scopul de a evita sau de a reduce fluctuațiile care apar în semnalele de intrare, cele provenite de la butoane. Fluctuațiile pot fi rezultatul contactului instabil între două suprafețe metalice în momentul apăsării butonului, și pot cauza erori sau probleme în sistemul electronic care utilizează acel semnal de intrare.</p>
	<p>Divizorul de frecvență are rolul de a împărți frecvența semnalului de intrare la o frecvență mai mică pe semnalul de ieșire. Divizorul de frecvență este utilizat pentru a obține un semnal cu o frecvență redusă în comparație cu semnalul de intrare, de obicei pentru a sincroniza și a controla temporizările într-un sistem digital.</p>
	<p>Scopul componentei "delayer" este de a introduce un anumit timp de întârziere (delay) și de a genera un semnal de ieșire pe logică 1 după acea perioadă de timp. Este utilizat pentru a asigura sincronizarea dintre clock și enable-urile din UC cu toate RAM-urile.</p>
	<p>Această componentă ajută utilizatorul să formeze suma cu ușurință folosind butonul de adăugare a bancnotei și "bancnota" pentru a selecta tipul de bancnotă. Componenta permite utilizatorului să introducă succesiv bancnote de diferite valori și să calculeze suma totală a acestora. Butonul „ok” confirmă suma finală.</p>
	<p>Componenta este responsabilă de decodificarea informației transmise de către utilizator sau de RAM. Aceasta preia informațiile referitoare la tipul de bancnotă selectată și numărul de apăsări ale butonului de adăugare a bancnotei (vector de 12 biți) și le convertește într-o sumă corespunzătoare.</p>
	<p>Componenta o unitate de memorie specializată proiectată pentru a stoca și actualiza numărul de bancnote codificată pe care le deține o bancă într-un vector de 12 biți. Aceasta utilizează o arhitectură de memorie RAM pentru a permite înregistrarea și gestionarea eficientă a cantității de bancnote disponibile. Componenta oferă operații de citire și scriere, permițând bancomatului să acceseze și să actualizeze numărul de bancnote.</p>

 <p>The diagram shows a component named citire_PIN. It has five input ports on the left: <i>cifra1</i>, <i>cifra2</i>, <i>cifra3</i>, <i>cifra4</i>, and <i>confirm_pin</i>. It has one output port on the right labeled <i>output</i>.</p>	<p>Componenta facilitează procesul de formare a PIN-ului utilizatorului. Aceasta oferă un mecanism prin care utilizatorul poate selecta și introduce cifrele care alcătuiesc PIN-ul său. Utilizatorul poate alege și introduce succesiv cifrele PIN-ului prin intermediul interfeței oferite de componentă. Fiecare cifră poate fi selectată dintr-un set predefinit de la 0 până la 3.</p>
 <p>The diagram shows a component named comparator_10. It has three input ports on the left: <i>enable</i>, <i>A</i>, and <i>B</i>. It has one output port on the right labeled <i>posibil</i>.</p>	<p>Componenta este o unitate specializată în cadrul unui sistem bancar care are rolul de a compara două sume de 10 biți: suma decodificată din contul utilizatorului și suma pe care acesta dorește să o retragă. Dacă suma din contul utilizatorului este mai mare decât suma ce dorește să o retragă, atunci semnalul <i>posibil</i> se activează.</p>
 <p>The diagram shows a component named RAM_PIN. It has four input ports on the left: <i>clk</i>, <i>enable</i>, <i>mode</i>, and <i>input</i>. It has two output ports on the right: <i>finded</i> and <i>position</i>.</p>	<p>Componenta este un sistem de securitate care compară toate PIN-urile stocate în memoria RAM cu PIN-ul introdus de către utilizator. Dacă componenta găsește pinul, atunci se activează semnalul “găsit” pentru UC și poziția acestui pentru “RAM_sold”. Această componentă facilitează și de opțiunea de schimbare a PIN-ului.</p>
 <p>The diagram shows a component named RAM_sold_cont. It has five input ports on the left: <i>clk</i>, <i>enable</i>, <i>adresa_ram</i>, <i>load</i>, and <i>mode</i>. It has one output port on the right labeled <i>output</i>.</p>	<p>Această componentă reține suma de bani pentru fiecare utilizator abonat la bancomat. Atunci când se efectuează operații de retragere sau adăugare a soldului, componenta permite actualizarea sumei de bani înregistrată în contul utilizatorului respectiv.</p>
 <p>The diagram shows a component named scazator_retragere. It has four input ports on the left: <i>enable</i>, <i>scz_banc</i>, <i>scz_cont...</i>, and <i>scz_de_sc...</i>. It has three output ports on the right: <i>out_RAM_banc</i>, <i>out_RAM_cont</i>, and <i>succes</i>.</p>	<p>Componenta are rolul de a efectua retragerea de bancnote pentru un utilizator, verificând în prealabil disponibilitatea acestora și încercând să formeze suma solicitată din bancnote mai mici utilizând un algoritm “greedy”, în cazul în care nu sunt suficiente. În urma retragerii cu succes, se actualizează numărul de bancnote rămase atât pentru utilizator, cât și pentru bancă, și se activează un semnal corespunzător. În caz contrar, dacă nu este posibilă retragerea, nu se activează niciun semnal.</p>

 <p>enable out_RAM_banc sum_cont out_RAM_util sum_bancnote succes sum_adunat sumator_depunere</p>	<p>Atunci când se efectuează operațiunea de adăugare a bancnotelor, componenta înregistrează suma adăugată în contul utilizatorului, crește numărul de bancnote disponibile în bancomat și actualizează informațiile corespunzătoare. Dacă operația a fost efectuată cu succes, atunci se activează semnalul "succes".</p>
 <p>clk output_af.. enable output_mux input afisor_structural</p>	<p>Componenta menționată, numită "AfisareSSD", are rolul de a afișa informații pe un SSD (Display cu 7 segmente) conectat la placa FPGA. Pentru a realiza acest lucru, componenta utilizează un divizor de frecvență și o ieșire pentru anoduri.</p> <p>Divizorul de frecvență este utilizat pentru a genera un semnal de frecvență redusă, care este necesar pentru a controla afișajul pe SSD. Acesta împarte frecvența de intrare la un anumit raport, astfel încât afișarea să se realizeze într-un mod vizibil pentru ochiul uman.</p>
 <p>Input1 Output Input2 SAU_12Biti</p> <p>Input1 Output Input2 SAU_1Bit</p>	<p>Aceste 2 componente funcționează ca o poarta "SAU" pe o legătură de biți diferiți. Acestea comuta într-un mod eficient între rezultatele de la operațiile "retrage" și "adăugare" a sumei.</p>
 <p>input output not1</p>	<p>Această componentă simulează o poartă "NOT" pentru a ajuta UC-ul să efectueze operația "Schimbare PIN".</p>

Codul din spatele fiecărei resurse:

- Debouncer:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY debouncer IS
  PORT (buton, clk: in std_logic;
        iesire: out std_logic);
END debouncer;

ARCHITECTURE TypeArchitecture OF debouncer IS
BEGIN
  process(buton, clk)
    variable num: integer:=0;
  begin
    if(buton='1') then
      if(rising_edge(clk)) then num:=num+1;
      end if;
    else num:=0;
    end if;

    if(num>10) then iesire<='1';
    else iesire<='0';
    end if;
  end process;
END TypeArchitecture;
```

- Divizor de frecvență:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY div_frecv IS
  PORT (clk : in std_logic;
        clk_out : out std_logic);
END div_frecv;

ARCHITECTURE arh_cnt OF div_frecv IS
  signal clk_div: std_logic := '0';
  signal cnt: integer := 0;
BEGIN
```

```

process(clk, cnt)
  BEGIN
    if(clk = '1' and clk'event) then
      if(cnt = 2) then
        cnt <= 1;
        clk_div <= not clk_div;
      else
        cnt <= cnt+1;
      end if;
    end if;

```

- Delayer:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY delayer IS
  PORT (Input: in std_logic;
        Output: out std_logic);
END delayer;

ARCHITECTURE TypeArchitecture OF delayer IS
BEGIN
  process(Input)
  begin
    if (Input = '1') then
      Output<='1' after 100 ns;
    else
      Output<='0' after 100 ns;
    end if;

    end process;
END TypeArchitecture;

```

- Citire bancnote:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY citire_bancnote IS
  PORT (button: in std_logic;
        bancnota: in std_logic_vector(1 downto 0);
        output : out std_logic_vector(11 downto 0);
        ok : in std_logic);
END citire_bancnote;

```



```

ARCHITECTURE arh1 OF citire_bancnote IS
signal out_cod: std_logic_vector(11 downto 0) := "000000000000";
BEGIN
  process(bancnota, ok, button)
    variable cnt500: integer := 0;
    variable cnt100: integer := 0;
    variable cnt50: integer := 0;
    variable cnt10: integer := 0;
  begin
    case bancnota is
      when "00" => if(button = '1' and button'event) then
        cnt10:=cnt10+1;
        out_cod(2 downto 0)<= std_logic_vector(to_unsigned(cnt10, 3));
      end if;
      when "01" => if(button = '1' and button'event) then
        cnt50:=cnt50+1;
        out_cod(5 downto 3)<= std_logic_vector(to_unsigned(cnt50, 3));
      end if;
      when "10" => if(button = '1' and button'event) then
        cnt100:=cnt100+1;
        out_cod(8 downto 6)<= std_logic_vector(to_unsigned(cnt100, 3));
      end if;
      when others => if(button = '1' and button'event) then
        cnt500:=cnt500+1;
        out_cod(11 downto 9)<= std_logic_vector(to_unsigned(cnt500, 3));
      end if;
    end case;
    if(ok = '1') then
      output<=out_cod;
    else
      output<="000000000000";
    end if;
  end process;
END arh1;

```

- Convertor sumă:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY convertor_suma IS
  PORT (input : in std_logic_vector(11 downto 0);
        output : out std_logic_vector(9 downto 0));
END convertor_suma;

ARCHITECTURE TypeArchitecture OF convertor_suma IS
BEGIN
  process(input)
    variable sum : integer := 0;

    begin
      sum:=to_integer(unsigned(input(11 downto 9)))*500 +
      to_integer(unsigned(input(8 downto 6)))*100 + to_integer(unsigned(input(5
downto 3)))*50 + to_integer(unsigned(input(2 downto 0)))*10;
      output<= std_logic_vector(to_unsigned(sum, 10));
    end process;
END TypeArchitecture;

```

- RAM bancnote:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE ieee.std_logic_unsigned.all;

ENTITY RAM_sold_bancnote IS
  PORT (clk, enable : in std_logic;
        load : in std_logic_vector(11 downto 0);
        mode : in std_logic;  -- 0 => read; 1 => write
        output : out std_logic_vector(11 downto 0));
END RAM_sold_bancnote;

ARCHITECTURE arh OF RAM_sold_bancnote IS
  type matrice is array (0 to 3) of std_logic_vector(2 downto 0);
BEGIN

```

```

process(load, mode, enable, clk)
  variable mem : matrice := ("001", "010", "000", "011");
  variable aux : std_logic_vector(11 downto 0);
begin
  if(clk = '1' and clk'event) then
    if(enable = '1') then
      if(mode = '0') then  --read
        aux(11 downto 9) := mem(0);
        aux(8 downto 6) := mem(1);
        aux(5 downto 3) := mem(2);
        aux(2 downto 0) := mem(3);
      elsif (mode = '1') then --write
        mem(0) := load(11 downto 9);
        mem(1) := load(8 downto 6);
        mem(2) := load(5 downto 3);
        mem(3) := load(2 downto 0);
      end if;
      output <= aux;
    end if;
  end if;
end process;

```

END arh;

- Citire PIN:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY citire_PIN IS
  PORT (cifra1, cifra2, cifra3, cifra4 : in std_logic_vector(1 downto 0);
    confirm_pin : in std_logic;
    output : out std_logic_vector(7 downto 0));
END citire_PIN;

ARCHITECTURE arh OF citire_PIN IS
BEGIN
  process(cifra1, cifra2, cifra3, cifra4, confirm_pin)
    variable val : std_logic_vector(7 downto 0);
  begin

    val(7 downto 6) := cifra1;
    val(5 downto 4) := cifra2;
    val(3 downto 2) := cifra3;
    val(1 downto 0) := cifra4;

```

```

if(confirm_pin = '1') then
    output <= val;
end if;
end process;
END arh;

```

- Comparator pe 10 biți:

```

LIBRARY ieee;
USE ieee.std_l
ogic_1164.all;

ENTITY comparator_10 IS
    PORT (enable: in std_logic;
        A, B: in std_logic_vector(9 downto 0);
        posibil: out std_logic);
END comparator_10;

ARCHITECTURE Arch_comp_10 OF comparator_10 IS
BEGIN
    process(A, B, enable)
    begin
        posibil<='0';
        if(enable = '1') then
            if(A > B) then
                posibil<='0';
            elsif (A = B) then
                posibil<='1';
            else
                posibil<='1';
            end if;
        end if;
    end process;
END Arch_comp_10;

```

- RAM PIN

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE ieee.std_logic_unsigned.all;

ENTITY RAM_PIN IS
    PORT (clk, enable, mode: in std_logic;  -- switch
        input: in std_logic_vector(7 downto 0); --citire ram
        finded: out std_logic;
        position: out std_logic_vector(1 downto 0));
END RAM_PIN;

```

```

ARCHITECTURE arch OF RAM_PIN IS
  type matrice is array(0 to 3) of std_logic_vector(7 downto 0);
BEGIN
  process(mode, clk, input, enable)
    variable mem: matrice:=("00110110", "10000111", "11111010", "10110001");
    variable pozitie_veche: integer :=100;
    variable contor: integer :=0;
    variable succesful: std_logic :='0';
  begin
    if(enable = '1') then
      if(clk'event and clk='1') then
        succesful := '0';
        if(mode = '0') then
          for i in 0 to 3 loop
            if(mem(i) = input) then
              succesful:='1';
              pozitie_veche := i;
            end if;
          end loop;
        elsif(mode = '1') then
          mem(pozitie_veche) := input; --scriere
          succesful := '0';
        end if;
      end if;
    end if;
    finded <= succesful;
    position <= std_logic_vector(to_unsigned(pozitie_veche, 2));
  end process;
END arch;

```

- RAM sold conturi:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
USE ieee.std_logic_unsigned.all;

ENTITY RAM_sold_cont IS
  PORT (clk, enable : in std_logic;
    adres_ram: in std_logic_vector(1 downto 0);
    load: in std_logic_vector(11 downto 0);
    mode: in std_logic; --0 => read, 1 => write
    output: out std_logic_vector(11 downto 0));
END RAM_sold_cont;

```

```

ARCHITECTURE TypeArchitecture OF RAM_sold_cont IS
    type matrice is array (0 to 3) of std_logic_vector(11 downto 0);
BEGIN
    process(adress_ram, load, mode, enable, clk)
        variable mem : matrice := ("000001010000", "001011000100",
"000010000001", "001011001001");
        begin
            if(clk = '1' and clk'event) then
                if(enable = '1') then
                    if(mode = '0') then
                        output <= mem(conv_integer(adress_ram)); --read
                    elsif (mode = '1') then
                        mem(conv_integer(adress_ram)) := load; --write
                    end if;
                end if;
            end if;
        end process;
    END TypeArchitecture;

```

- Scăzător retragere numerar:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY scazator_retragere IS
    PORT (enable: in std_logic;
        scz_banc: in std_logic_vector(11 downto 0);
        scz_cont_util: in std_logic_vector(11 downto 0);
        scz_de_scazut: in std_logic_vector(11 downto 0);
        out_RAM_banc: out std_logic_vector(11 downto 0);
        out_RAM_cont: out std_logic_vector(11 downto 0);
        succes: out std_logic);
END scazator_retragere;

ARCHITECTURE TypeArchitecture OF scazator_retragere IS
BEGIN
    process(enable, scz_banc, scz_de_scazut)
        variable var_500: integer := 0;
        variable var_100: integer := 0;
        variable var_50: integer := 0;
        variable var_10: integer := 0;
        variable cmp2, cmp1: integer:=0;
        variable ok: std_logic:= '1';
        variable sum: std_logic_vector(11 downto 0):="000000000000";
    
```

```

begin
  out_RAM_banc <="000000000000";
  out_RAM_cont <="000000000000";
  succes <= '0';
  if(enable = '1') then
    cmp2:=to_integer(unsigned(scz_banc(11 downto 9)))*500 +
to_integer(unsigned(scz_banc(8 downto 6)))*100 + to_integer(unsigned(scz_banc(5
downto 3)))*50 + to_integer(unsigned(scz_banc(2 downto 0))*10;
    cmp1:=to_integer(unsigned(scz_de_scazut(11 downto 9)))*500 +
to_integer(unsigned(scz_de_scazut(8 downto 6)))*100 +
to_integer(unsigned(scz_de_scazut(5 downto 3)))*50 +
to_integer(unsigned(scz_de_scazut(2 downto 0))*10;

    if(cmp2 >= cmp1) then -- cand suma adunata in RAM_banc este mai mare
decat ce cere utilizatorul
      succes<='1';
      var_500:=to_integer(unsigned(scz_banc(11 downto 9)))-
to_integer(unsigned(scz_de_scazut(11 downto 9)));
      var_100:=to_integer(unsigned(scz_banc(8 downto 6)))-
to_integer(unsigned(scz_de_scazut(8 downto 6)));
      var_50:=to_integer(unsigned(scz_banc(5 downto 3)))-
to_integer(unsigned(scz_de_scazut(5 downto 3)));
      var_10:=to_integer(unsigned(scz_banc(2 downto 0)))-
to_integer(unsigned(scz_de_scazut(2 downto 0)));

      for i in 0 to 6 loop
        if(var_500 < 0) then
          var_100:=var_100 - 5;
          var_500:=var_500 + 1;
        end if;
      end loop;
      if(var_500 < 0) then
        ok := '0';
      end if;

      for i in 0 to 6 loop
        if(var_100 < 0) then
          var_100:=var_100 + 1;
          var_50:=var_50 - 2;
        end if;
      end loop;
      if(var_100 < 0) then
        ok := '0';
      end if;

```

```

for i in 0 to 6 loop
    if(var_50 < 0) then
        var_50:=var_50 +1;
        var_10:=var_10 - 5;
    end if;
end loop;
if(var_50 < 0) then
    ok := '0';
end if;

if(var_10 < 0) then
    ok := '0';
end if;

sum(11 downto 9):= std_logic_vector(to_unsigned(var_500, 3));
sum(8 downto 6):= std_logic_vector(to_unsigned(var_100, 3));
sum(5 downto 3):= std_logic_vector(to_unsigned(var_50, 3));
sum(2 downto 0):= std_logic_vector(to_unsigned(var_10, 3));

if(ok='1') then
    out_RAM_banc<= sum;
    succes<='1';

    var_500:=to_integer(unsigned(scz_cont_util(11 downto 9)))-
to_integer(unsigned(scz_de_scazut(11 downto 9)));
    var_100:=to_integer(unsigned(scz_cont_util(8 downto 6)))-
to_integer(unsigned(scz_de_scazut(8 downto 6)));
    var_50:=to_integer(unsigned(scz_cont_util(5 downto 3)))-
to_integer(unsigned(scz_de_scazut(5 downto 3)));
    var_10:=to_integer(unsigned(scz_cont_util(2 downto 0)))-
to_integer(unsigned(scz_de_scazut(2 downto 0)));

    sum(11 downto 9):= std_logic_vector(to_unsigned(var_500, 3));
    sum(8 downto 6):= std_logic_vector(to_unsigned(var_100, 3));
    sum(5 downto 3):= std_logic_vector(to_unsigned(var_50, 3));
    sum(2 downto 0):= std_logic_vector(to_unsigned(var_10, 3));
    out_RAM_cont<=sum;

```



```

else
    out_RAM_banc <=scz_banc;
    out_RAM_cont <=scz_cont_util;
    succes<='0';
end if;
else
    out_RAM_banc <= scz_banc;
    out_RAM_banc<=scz_cont_util;
    succes<='0';
end if;
end if;
end process;
END TypeArchitecture;

```

- Sumator pentru depunere bancnote:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

USE ieee.numeric_std.all;
ENTITY sumator_depunere IS
    PORT (enable: in std_logic;
        sum_cont: in std_logic_vector(11 downto 0);
        sum_bancnote: in std_logic_vector(11 downto 0);
        sum_adunat: in std_logic_vector(11 downto 0);
        out_RAM_banc: out std_logic_vector(11 downto 0);
        out_RAM_util: out std_logic_vector(11 downto 0);
        succes: out std_logic);
END sumator_depunere;

ARCHITECTURE TypeArchitecture OF sumator_depunere IS
BEGIN
    process(enable, sum_cont, sum_bancnote, sum_adunat)
        variable var_500: integer := 0;
        variable var_100: integer := 0;
        variable var_50: integer := 0;
        variable var_10: integer := 0;
        variable sum: std_logic_vector(11 downto 0):="000000000000";
    begin
        succes <= '0';
        if(enable = '1') then
            var_500:=to_integer(unsigned(sum_cont(11 downto 9)))+
to_integer(unsigned(sum_adunat(11 downto 9)));
            var_100:=to_integer(unsigned(sum_cont(8 downto 6)))+
to_integer(unsigned(sum_adunat(8 downto 6)));

```

```

    var_50:=to_integer(unsigned(sum_cont(5 downto 3)))+
to_integer(unsigned(sum_adunat(5 downto 3)));
    var_10:=to_integer(unsigned(sum_cont(2 downto 0)))+
to_integer(unsigned(sum_adunat(2 downto 0)));
    sum(11 downto 9):= std_logic_vector(to_unsigned(var_500, 3));
    sum(8 downto 6):= std_logic_vector(to_unsigned(var_100, 3));
    sum(5 downto 3):= std_logic_vector(to_unsigned(var_50, 3));
    sum(2 downto 0):= std_logic_vector(to_unsigned(var_10, 3));

    out_RAM_util<=sum;

    var_500:=to_integer(unsigned(sum_bancnote(11 downto 9)))+
to_integer(unsigned(sum_adunat(11 downto 9)));
    var_100:=to_integer(unsigned(sum_bancnote(8 downto 6)))+
to_integer(unsigned(sum_adunat(8 downto 6)));
    var_50:=to_integer(unsigned(sum_bancnote(5 downto 3)))+
to_integer(unsigned(sum_adunat(5 downto 3)));
    var_10:=to_integer(unsigned(sum_bancnote(2 downto 0)))+
to_integer(unsigned(sum_adunat(2 downto 0)));
    sum(11 downto 9):= std_logic_vector(to_unsigned(var_500, 3));
    sum(8 downto 6):= std_logic_vector(to_unsigned(var_100, 3));
    sum(5 downto 3):= std_logic_vector(to_unsigned(var_50, 3));
    sum(2 downto 0):= std_logic_vector(to_unsigned(var_10, 3));

    out_RAM_banc<=sum;
    succes <= '1';
else
    out_RAM_banc<="000000000000";
    out_RAM_util<="000000000000";
    succes <= '0';
end if;

end process;
END TypeArchitecture;

```

- Afișor pentru SSD:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY afisor_structural IS
  PORT (clk, enable : in std_logic;
    input : in std_logic_vector(9 downto 0);
    output_afisor : out std_logic_vector(6 downto 0);
    output_mux : out std_logic_vector(3 downto 0));
END afisor_structural;

ARCHITECTURE arh_afisor_struct OF afisor_structural IS

  component counter_afisor_div IS
    PORT (
      clk , enable: in std_logic;
      out_cnt : out std_logic_vector(16 downto 0)
    );
  END component;

  component dcd_suma_afisor IS
    PORT (
      input : in std_logic_vector(9 downto 0);
      unit, zeci, sute, mii : out std_logic_vector(3 downto 0)
    );
  END component;

  component mux_anod_afisor IS
    PORT (
      sel : in std_logic_vector(1 downto 0);
      output: out std_logic_vector(3 downto 0)
    );
  END component;

  component mux_cifra_afisor IS
    PORT (
      input0, input1, input2, input3 : in std_logic_vector(3 downto 0);
      sel : in std_logic_vector(1 downto 0);
      output : out std_logic_vector(3 downto 0)
    );
  END component;
```

```

component dcd_hexa_7seg_afisor IS
  PORT (
    input : in   std_logic_vector(3 downto 0);
    output : out std_logic_vector(6 downto 0)
  );
END component;

signal s_unit, s_zeci, s_sute, s_mii : std_logic_vector(3 downto 0);
signal s_out_cnt : std_logic_vector(16 downto 0);
signal s_output_cifra_afisor : std_logic_vector(3 downto 0);
BEGIN
  p_counter_afisor : counter_afisor_div port map (clk, enable, s_out_cnt);

  p_dcd_suma_afisor : dcd_suma_afisor port map (input, s_unit, s_zeci, s_sute,
s_mii);
  p_mux_cifra_afisor : mux_cifra_afisor port map (s_unit, s_zeci, s_sute, s_mii,
s_out_cnt(1 downto 0), s_output_cifra_afisor);
  p_dcd_hexa_7seg_afisor : dcd_hexa_7seg_afisor port map (s_output_cifra_afisor,
output_afisor);
  p_mux_anod_afisor : mux_anod_afisor port map (s_out_cnt(1 downto 0),
output_mux);
END arh_afisor_struct;

-----

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY counter_afisor_div IS
  PORT (clk , enable: in std_logic;
    out_cnt : out std_logic_vector(16 downto 0));
END counter_afisor_div;

ARCHITECTURE arh_cnt OF counter_afisor_div IS

  signal cnt : std_logic_vector(16 downto 0) := "0000000000000000";
  signal div_clk: std_logic := '0';
  signal nr_div: integer := 1;
BEGIN
  process(clk, nr_div, enable)
  BEGIN
    if(enable = '1') then
      if(clk = '1' and clk'event) then
        if(nr_div = 4) then
          nr_div <= 1;

```

```

elsif(nr_div = 2) then
    div_clk <= '0';
    nr_div <= nr_div + 1;
else
    div_clk <= '1';
    nr_div <= nr_div + 1;
end if;
end if;
end if;
end process;

process(div_clk, enable)
begin
    if(enable = '1') then
        if(div_clk = '1' and div_clk'event) then
            cnt <= cnt + 1;
        end if;
    end if;
end process;
    out_cnt <= cnt;
END arh_cnt;

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use ieee.numeric_std.all;

ENTITY dcd_suma_afisor IS
    PORT (
        input : in std_logic_vector(9 downto 0);
        unit, zeci, sute, mii : out std_logic_vector(3 downto 0)
    );
END dcd_suma_afisor;

ARCHITECTURE arh_dcd OF dcd_suma_afisor IS
    signal a : natural;
BEGIN
    a <= conv_integer(input);
    unit <= std_logic_vector(to_unsigned(a rem 10, 4));
    zeci <= std_logic_vector(to_unsigned(a/10 rem 10, 4));
    sute <= std_logic_vector(to_unsigned(a/100 rem 10, 4));
    mii <= std_logic_vector(to_unsigned(a/1000 rem 10, 4));
END arh_dcd;

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux_anod_afisor IS
  PORT (sel : in std_logic_vector(1 downto 0);
        output: out std_logic_vector(3 downto 0));
END mux_anod_afisor;

ARCHITECTURE arh_mux_anod OF mux_anod_afisor IS
BEGIN
  process(sel)
  begin
    case sel is
      when "00" => output <= "0001";
      when "01" => output <= "0010";
      when "10" => output <= "0100";
      when others => output <= "1000";
    end case;
  end process;
END arh_mux_anod;

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY mux_cifra_afisor IS
  PORT (input0, input1, input2, input3 : in std_logic_vector(3 downto 0);
        sel : in std_logic_vector(1 downto 0);
        output : out std_logic_vector(3 downto 0));
END mux_cifra_afisor;

ARCHITECTURE arh_mux_cifra OF mux_cifra_afisor IS
BEGIN

  process(sel, input0, input1, input2, input3)
  begin
    case sel is
      when "00" => output <= input0;
      when "01" => output <= input1;
      when "10" => output <= input2;
      when others => output <= input3;

    end case;
  end process;
END arh_mux_cifra;

```

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY dcd_hexa_7seg_afisor IS
    PORT (input : in std_logic_vector(3 downto 0);
        output : out std_logic_vector(6 downto 0));
END dcd_hexa_7seg_afisor;

ARCHITECTURE arh_7seg OF dcd_hexa_7seg_afisor IS
BEGIN
    output <= "111110" when input = x"0" else
        "0110000" when input = x"1" else
        "1101101" when input = x"2" else
        "1111001" when input = x"3" else
        "0110011" when input = x"4" else
        "1011011" when input = x"5" else
        "1011111" when input = x"6" else
        "1110000" when input = x"7" else
        "1111111" when input = x"8" else
        "1111011" when input = x"9" else
        "1110111" when input = x"A" else
        "0011111" when input = x"B" else
        "1001110" when input = x"C" else
        "0111101" when input = x"D" else
        "1001111" when input = x"E" else
        "1000111" when input = x"F";
END arh_7seg;

```

- Poartă SAU pe 12 biți:

```

LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SAU_12Biti IS
    PORT (Input1, Input2: in std_logic_vector(11 downto 0);
        Output: out std_logic_vector(11 downto 0));
END SAU_12Biti;

ARCHITECTURE TypeArchitecture OF SAU_12Biti IS
BEGIN
    Output<=Input1 or Input2;
END TypeArchitecture;

```

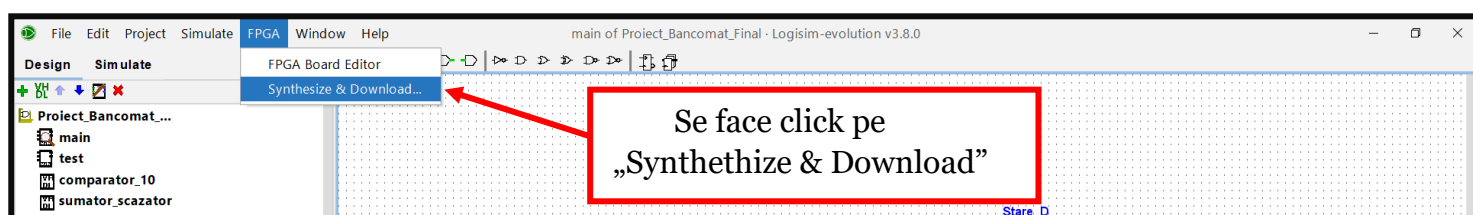
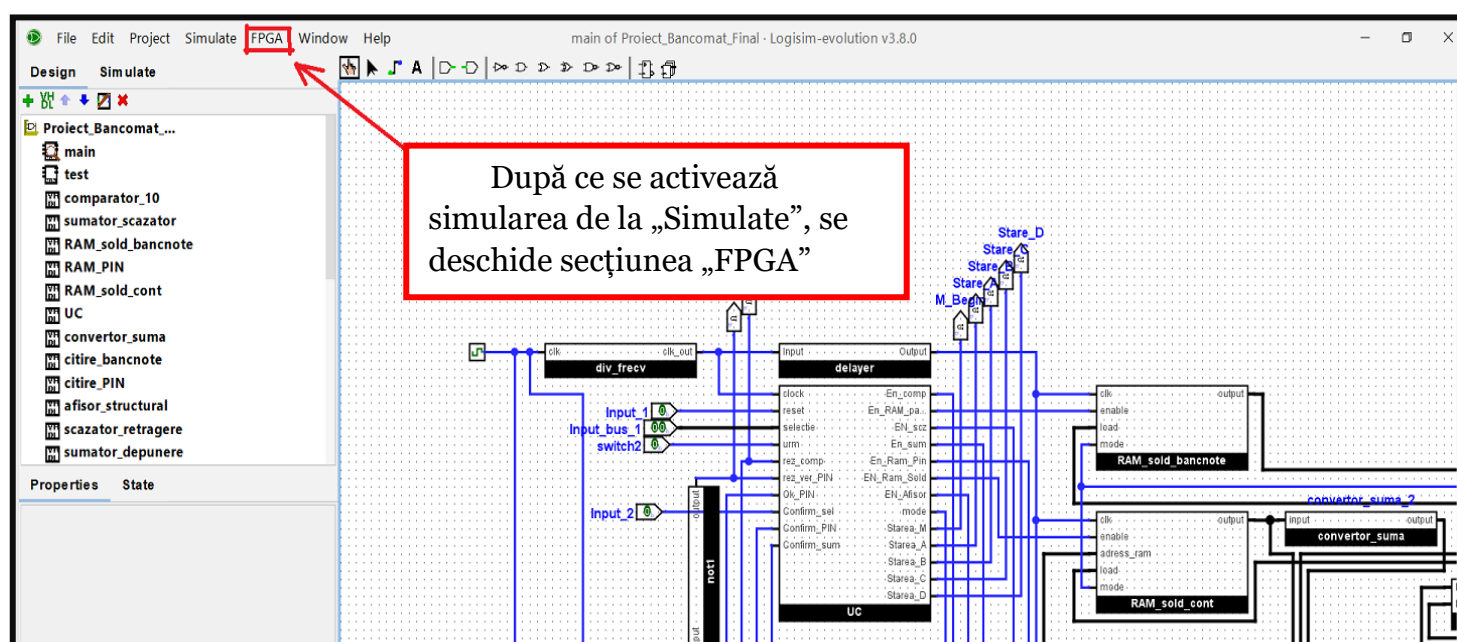
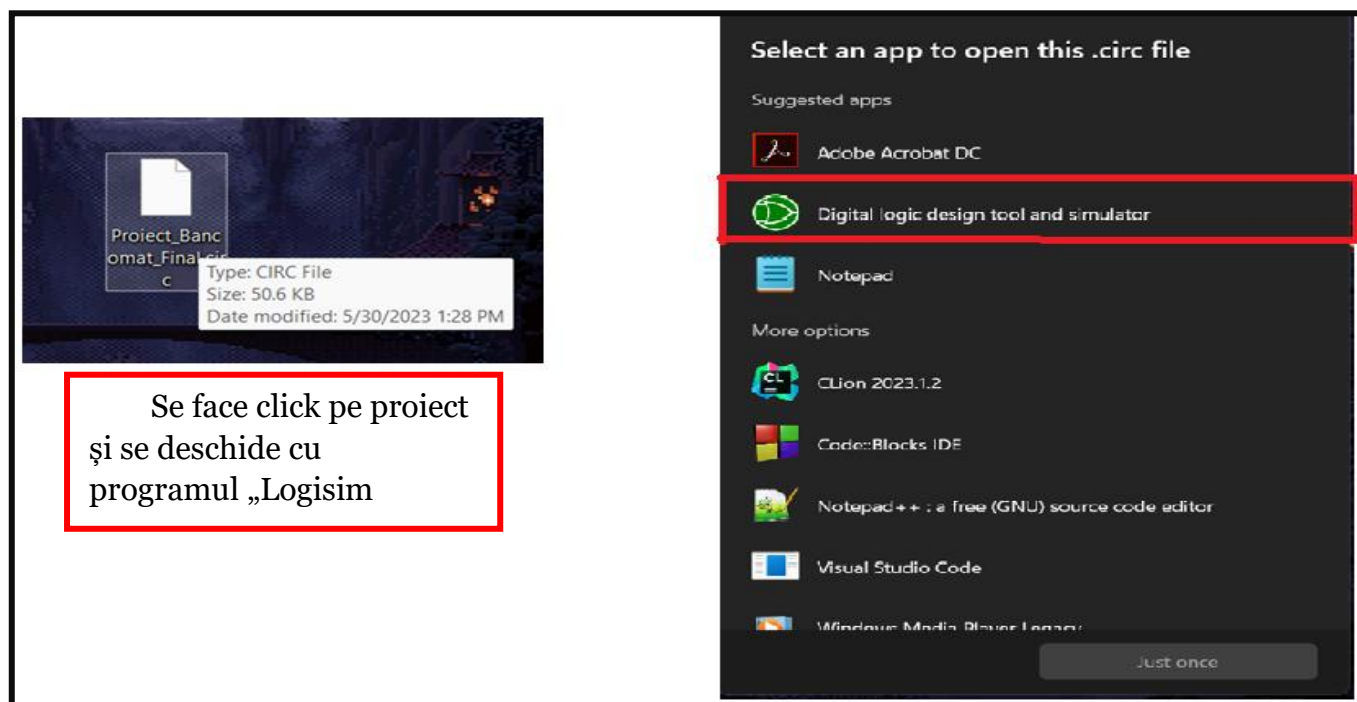
- Poartă SAU pe 1 bit:

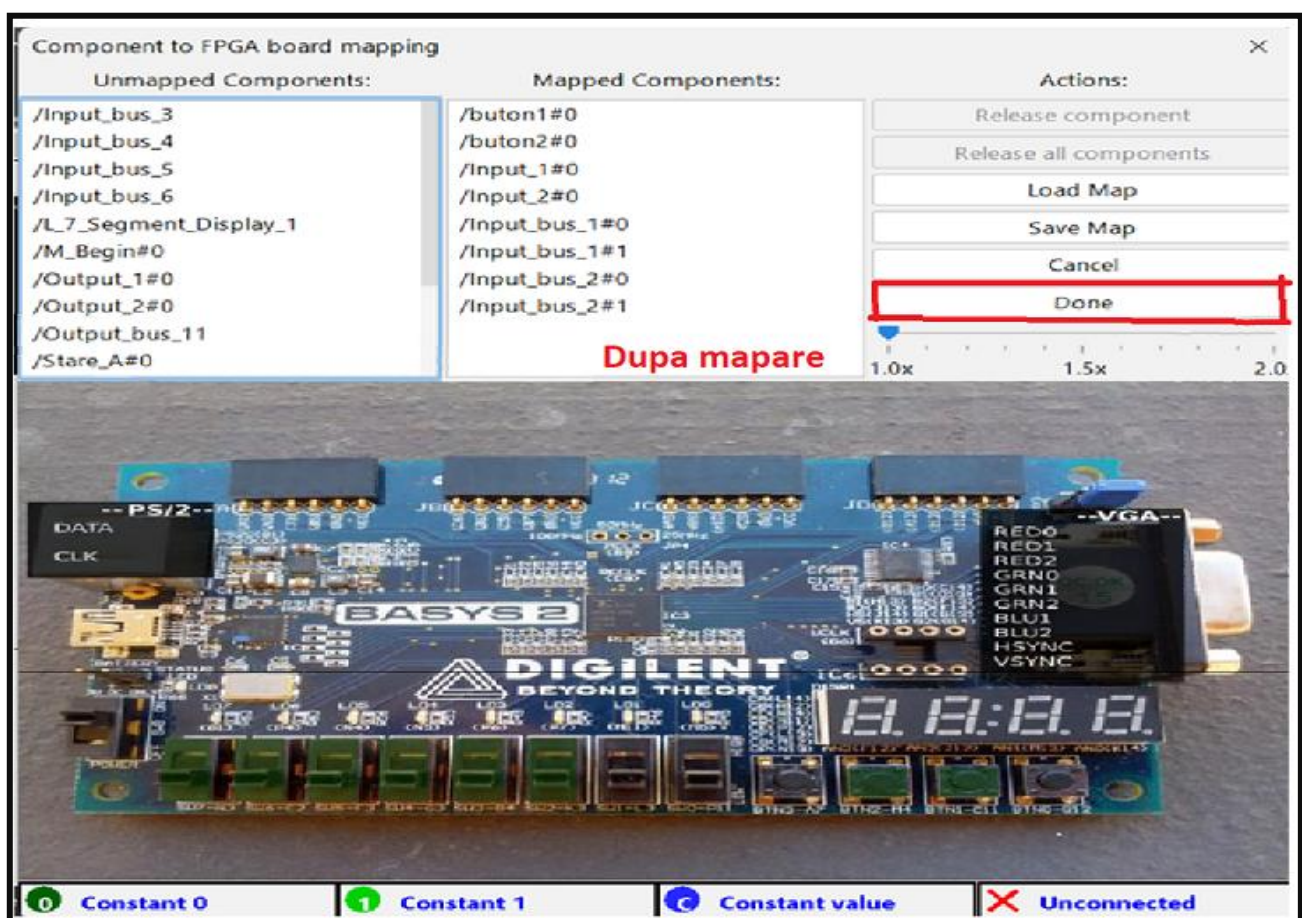
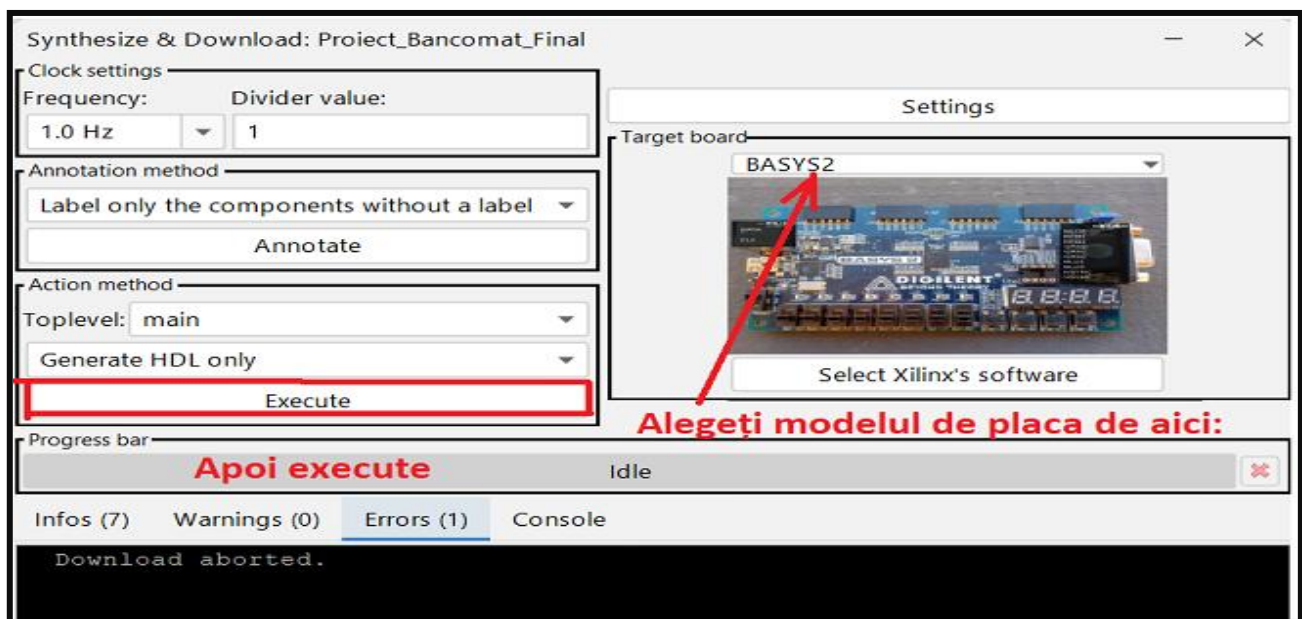
```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY SAU_1Bit IS  
  PORT (Input1, Input2: in std_logic;  
        Output: out std_logic);  
END SAU_1Bit;  
  
ARCHITECTURE TypeArchitecture OF SAU_1Bit IS  
BEGIN  
  
  Output<=Input1 or Input2;  
END TypeArchitecture;
```

- Poartă NOT pe 1 bit:

```
LIBRARY ieee;  
USE ieee.std_logic_1164.all;  
  
ENTITY not1 IS  
  PORT (input: in std_logic;  
        output: out std_logic);  
END not1;  
  
ARCHITECTURE TypeArchitecture OF not1 IS  
BEGIN  
  output<=not input;  
END TypeArchitecture;
```


Instrucțiuni de întreținere și utilizare:





Justificarea soluției alese:

Am ales aceasta soluție de implementare în concordanță cu adevărata utilizare a unui automat bancar, operațiile efectuate fiind printre cele existente în realitate, reușind însă simplificarea lor.

Pentru „interogare sold”, varianta utilizată asigură afișarea sumei de bani din RAM-ul pentru solduri, pe baza adresei la care s-a identificat PIN-ul introdus.

Operația de „retragere numerar” necesită introducerea unei sume de către utilizator, care va fi comparată cu suma de care acesta dispune în cont (tot pe baza PIN-ului), apoi se va efectua scăderea, dacă solicitantul dispune de fonduri suficiente și dacă există destule bancnote. La acest pas se scade suma dorită pe baza principiului „greedy”, împrumutat din programarea software, evitându-se astfel compararea exhaustivă.

La „depunere numerar”, operațiunea are loc asemănător cu cea descrisă anterior pentru retragere, singura diferență fiind că se va adăuga în cont suma introdusă de către utilizator. Aceasta soluție nu necesită operații dificile.

Pentru operațiunea de „schimbare PIN”, se așteaptă introducerea unui nou cod, după care acesta se compară cu toate codurile existente în memoria automatului. Dacă PIN-ul nu există, acesta se va memora la adresa curentă.

După finalizarea oricărei din cele patru operații prezentate mai sus, bancomatul are capacitatea de a efectua o altă operație, iar, dacă se dorește introducerea unui alt card (identificat după alt cod PIN), automatul are capacitatea de a se reseta.

Astfel, prin abordările prezentate anterior, considerăm ca soluția noastră este simplă de înțeles și utilizată, fiind și destul de eficientă.

Posibilități de dezvoltare ulterioare:

La momentul actual, automatul bancar dispune de patru operațiuni de bază, existând posibilitatea ca funcționalitățile să se extindă.

Spre exemplu, ar putea fi adăugată opțiunea de „transfer suma”, ce ar necesita atât introducerea contului unde se dorește transferul (care duce la apariția necesității unei alte memorii RAM, unde să fie stocate conturile), cât și a sumei de transferat.

O alta modalitate de dezvoltare ar fi opțiunea de schimb valutar, ce necesită extinderea memoriei RAM (pentru a reține soldul în lei și în alta valută, să zicem euro), cât și introducerea și memorarea cursului valutar după care se va face transferul.

În ceea ce privește interacțiunea utilizator-automat, ar putea fi aduse îmbunătățiri la operațiunea de „retragere numerar”, pentru ca aceasta nu se efectuează corect atunci când se solicită o suma formatată dintr-o sumă de bancnote care ar forma o bancnotă de sine stătătoare (de exemplu, atunci când se solicită două bancnote de 50 în loc de una de 100). În cazul prezentat, conform euristicii implementate la metoda greedy, nu există posibilitatea de a introduce astfel de sume, ci doar bancnote de sine stătătoare (chiar dacă s-ar fi putut forma dintr-o sumă).

Astfel, o îmbunătățire ulterioară ar fi tratarea acestor cazuri fie prin schimbarea strategiei de calcul, fie prin modificarea modalității de introducere a bancnotelor.

Bibliografie:

- The Bancomat problem: an example of resource allocation in a partitionable asynchronous system([PII: S0304-3975\(01\)00398-X \(sciencedirectassets.com\)](#))
- Hayne, R. (2007, June), Vhdl Projects To Reinforce Computer Architecture Classroom Instruction Paper presented at 2007 Annual Conference & Exposition, Honolulu, Hawaii. 10.18260/1-2—1676 (<https://peer.asee.org/vhdl-projects-to-reinforce-computer-architecture-classroom-instruction.pdf>)
- Automated Planning for Ubiquitous Computing ([CSUR4904-63 \(rug.nl\)](#))