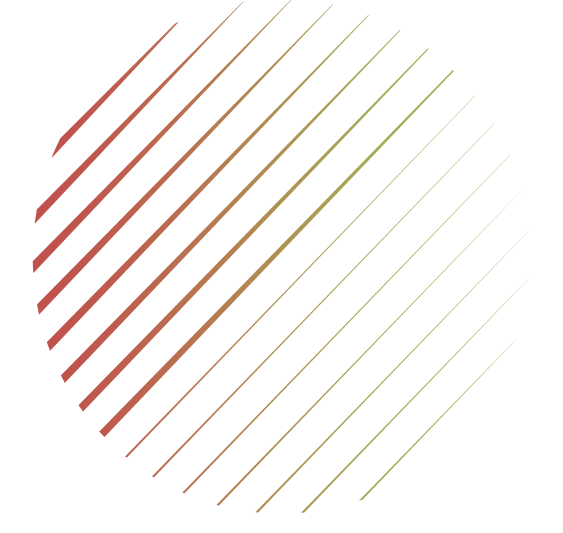
A picture containing logo, graphics, symbol, design

Description automatically generated



## Cotei Ruxanda-Maria &

## Grigoraș Victor-Andrei

## Coordonator: Maier Noema-Laura

**AUTOMAT BANCAR**

**31. 05. 2023**

Cuprins:

[Specificații: 3](#_Toc136446384)

[Proiectare și implementare: 3](#_Toc136446385)

[Inițializare: 4](#_Toc136446386)

[Configurare plăcuță FPGA: 4](#_Toc136446387)

[Schemă bloc: 4](#_Toc136446388)

[Unitate de comandă: 5](#_Toc136446389)

[Organigramă: 6](#_Toc136446390)

[Schema logică detaliată: 12](#_Toc136446391)

[Reprezentarea resurselor: 13](file:///C:\Users\rcote\Desktop\Documentatie%20bancomat.docx#_Toc136446392)

[Codul din spatele fiecărei resurse: 16](#_Toc136446393)

[Instrucțiuni de intreținere și utilizare: 34](#_Toc136446394)

[Justificarea soluției alese: 36](#_Toc136446395)

[Posibilități de dezvoltare ulterioare: 37](#_Toc136446396)

[Bibliografie: 38](#_Toc136446397)

Specificații:

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 existenta 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:

Înaintea începerii propriu-zise a interacțiunii dintre automat si utilizator, bancomatul este inițializat cu un număr cunoscut de bancnote de fiecare tip, cat si cu cele 4 PIN-uri după care se identifica 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 aflata pe cardul clientului, verificarea cu bancnotele din bancomat. Daca bancomatul are bancnotele necesare, se eliberează (se scad) acestea si se scade suma din cont cu suma introdusa. In caz contrar, se verifica daca se poate obține suma ceruta din alte bancnote, se procedează la fel ca in situația expusa anterior.

Operațiunea de depunere numerar consta in introducerea sumei de către utilizator, urmata de creșterea valorii din cont si a bancnotelor din casa automatului.

Operațiunea de schimbare PIN necesita introducerea unui nou cod PIN, verificarea ca acesta sa nu existe deja (nu pot fi doua PIN-uri identice) si schimbarea acestuia in memoria bancomatului.

Configurare plăcuță FPGA:

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

A picture containing electronics, circuit, electronic engineering, electronic component

Description automatically generated

Schemă bloc:

A picture containing text, screenshot, diagram, line

Description automatically generated 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.

A screenshot of a computer program

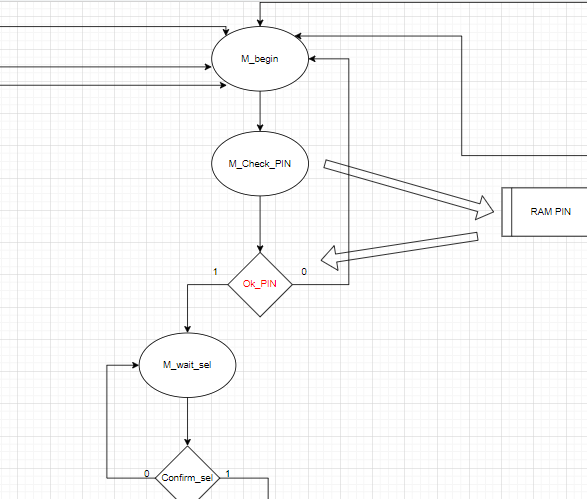
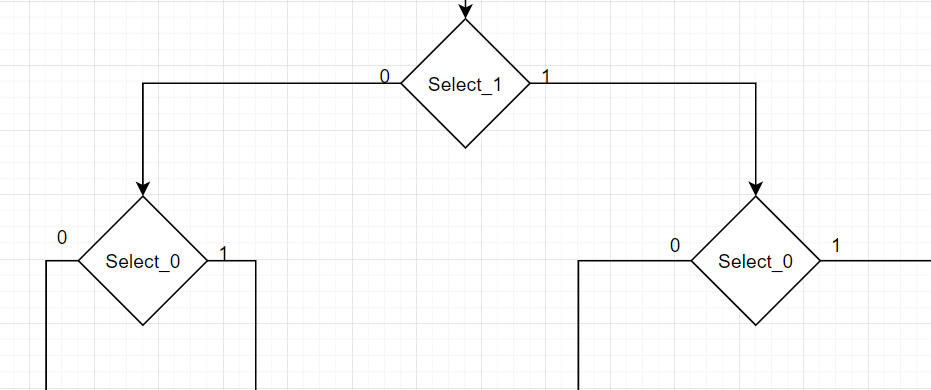
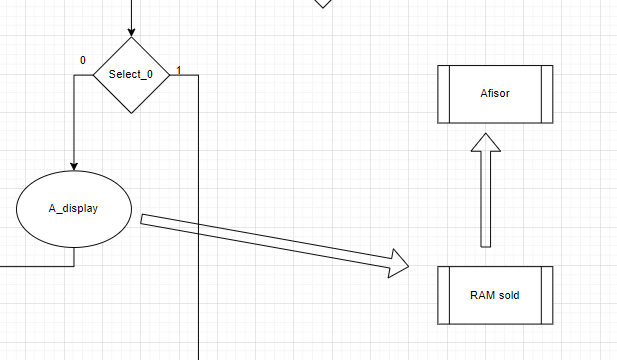
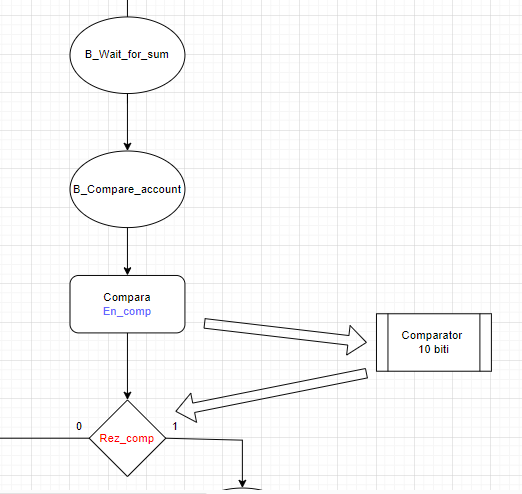
Description automatically generated with low confidence

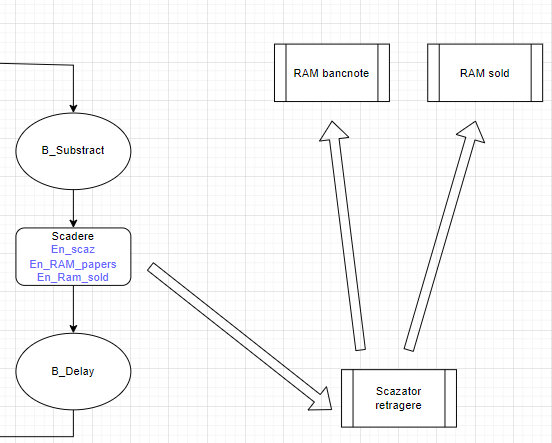
Unitatea de comandă este proiectată pe baza organigramei. Organigrama reprezintă o digramă de stări ce detaliază pas cu pas funcționalitatea automatului, pentru o mai bună înțelegere. Aceasta este alcătuită pe baza a trei componente simple: stare (oval), decizie (romb), ieșire (dreptunghi), la care s-au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

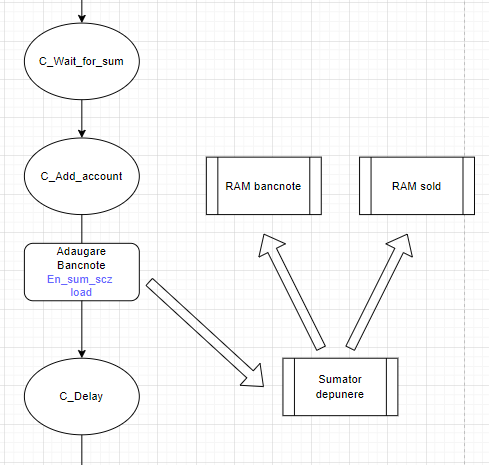
Organigrama proiectul este structurată în 5 categorii, în funcție de stări și operații, fiind codificate astfel:

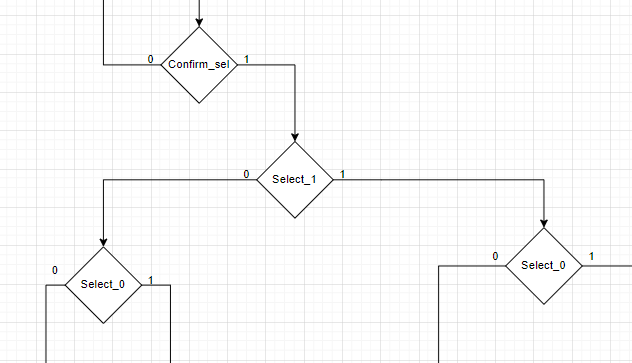
* M – inițializare și selectare operații;
* A – afișare sold;
* B – retragere numerar;
* C – depunere numerar;
* D – schimbare PIN.

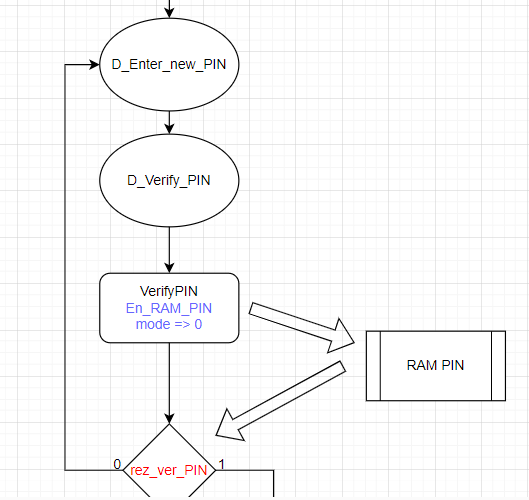
Organigramă:

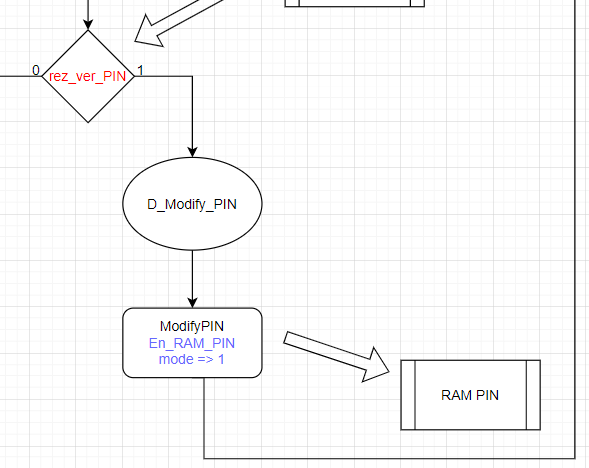
* Stările M:
* Selecțtii:
* Starea A:
* Stările B:

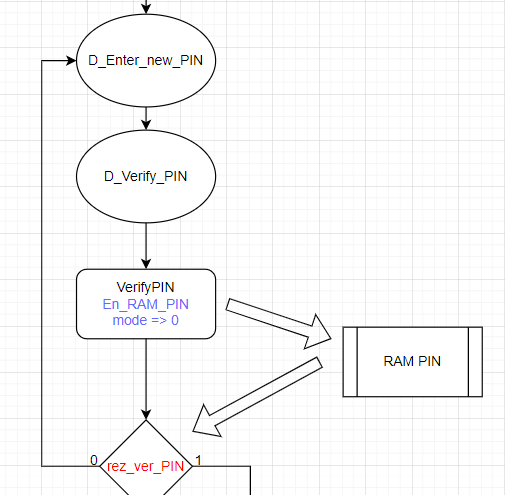


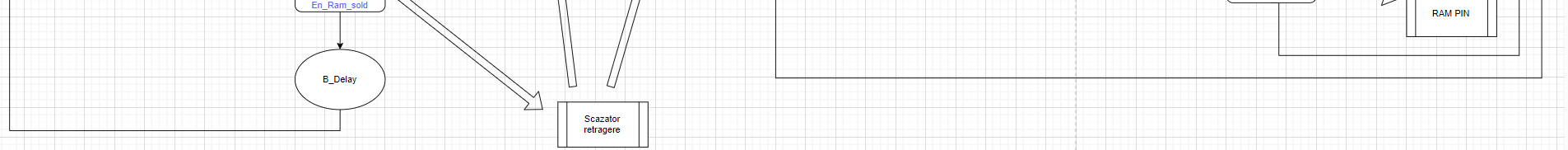
* Stările C:



* Stările D:





A picture containing diagram, text, line, plan

Description automatically generatedA picture containing line, diagram, text, plot

Description automatically generated

A picture containing text, diagram, plan, line

Description automatically generatedSchema logică detaliată:

A picture containing diagram, line, rectangle, screenshot

Description automatically generated

|  |  |
| --- | --- |
| A picture containing text, screenshot, line, font  Description automatically generated  Reprezentarea resurselor: | 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. |
|  | Divizorul de frecventa 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. |
|  | 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 si enable-urile din UC cu toate RAM-urile. |
|  | 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. |
|  | 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. |
|  | Componenta o unitate de memorie specializată proiectată pentru a stoca și actualiza numărul de bancnote codificata 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. |
|  | 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. |
|  | 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 sa retragă, atunci semnalul posibil se activează. |
|  | 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 si poziția acestui pentru “RAM\_sold”. Această componentă facilitează și de opțiunea de schimbare a PIN-ului. |
|  | 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. |
|  | 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. |
|  | 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”. |
|  | 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.  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. |
|  | Aceste 2 componente funcționează ca o poarta "SAU" pe o legătura de biți diferiți. Acestea comuta într-un mod eficient între rezultatele de la operațiile ”retrage” si ”adăugare” a sumei. |
|  | Această componentă simulează o poartă ”NOT” pentru a ajuta UC-ul să efectueze operația ”Schimbare PIN”. |

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**;  
 **end** **process**;  
 clk\_out <= clk\_div;  
**END** arh\_cnt;

* 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**;  
 adress\_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) := "00000000000000000";  
 **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 <= "1111110" **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;

A screenshot of a computer

Description automatically generatedInstrucțiuni de intreținere și utilizare:

Se face click pe proiect și se deschide cu programul „Logisim Evolution”

A computer screen shot of a computer

Description automatically generated with low confidence

După ce se activează simularea de la „Simulate”, se deschide secțiunea „FPGA”

A screenshot of a computer

Description automatically generated with medium confidence

Se face click pe „Synthethize & Download”

A screenshot of a computer

Description automatically generated

A picture containing text, electronics, screenshot, software

Description automatically generated

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, daca solicitantul dispune de fonduri suficiente și daca exista destule bancnote. La acest pas se scade suma dorita pe baza principiului „greedy”, împrumutat din programarea software, evitând-se astfel compararea exhaustiva.

La „depunere numerar”, operațiunea are loc asemănător cu cea descrisă anterior pentru retragere, singura diferență fiind că se va adauga în cont suma introdusă de către utilizator. Aceasta soluție nu necesita 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. Daca PIN-ul nu exista, acesta se va memora la adresa curentă.

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

Astfel, prin abordările prezentate anterior, consideram ca soluția noastră este simplu de înțeles si utilizat, fiind și destul de eficienta.

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 apăriția 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 ulterioara 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)](https://pdf.sciencedirectassets.com/271538/1-s2.0-S0304397500X04811/1-s2.0-S030439750100398X/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEC0aCXVzLWVhc3QtMSJIMEYCIQCEi%2B4Rcsd%2BTwr6UEPRR1ThxObDuAzbhLAT0BUgc6ryMgIhAI7R08mS4wNxwEz7bfPJBCrE5pF43XHOPXl1rAYdWkUPKrIFCGYQBRoMMDU5MDAzNTQ2ODY1IgwE%2BdOwlcgADTcpebsqjwWDAm53X3cXu68Cg3YNFdgRw%2By5%2F3xrHoLsWE%2FuQtLGp%2B7bizspHdFC06tbbhZvqbAquhMyRtgdmQCXaSzP3U0gNb6vYpHUVA9I1cL%2FiEpri3lEFu60ZATkwy1iLmCFRlrL6W0Ors2iL1DT5agIIzMZ5cKE7j%2FRU7EI%2Bpg2ZmZqSruyJoXt0fAiijw0S%2B%2BPw%2FNeXpiD0uZn3fo1x523mSdYsPhEV1c4a0SmB0%2F6VSIjpoTnSPWrR7B%2Bppwhmjp%2Fxag2DZ9pOOP628GJM3w5WYmrD5dgqoub8a%2Bt4W%2F2xMovHRaiyPU9JUjEvAsPpnwrAxdj6%2FCbBpPYMj6vwjybq4oblQQS3UDjrB6iduh7PcGKdSQvrw%2FhBDRB70mYixHo9bS6y7BXnmQyZ3VqNpDCuqSmwj7Hxs7dLqBLKzWgGt1vOgoXLklbsv2bOcYS3F%2B7TpZOXl%2F%2BPSBZYk0pns7nShC0VXo2ef%2BgtXGBjL2nnga3ydV9ZVE5KvKvUNSfKPU7pROF25sXTmjqeDs29aqiwihdlTmfUvvNaIe28BmL5RY8RGyj5uLLJR8L%2Bcm2mP12%2FnuGYE1cMNmE%2Fk5o3tR%2FG%2Fr8Mc3FqodPqRPnbHf7XBFHCAs3dELkW22ucUbJmtj4pzu5acM4%2F9Svqi1MRWlQ4jhm1Ki7hUx0ycKpWR5Wpv5vo0zk5phSnveUgAtQFO9Y24PAe8ZTtZV6Lo7y2wKUne2wJzDckME7lmxX8ZCvvmumwnyay%2B8nbptn3p0ynXuQyucY3z12tgJsbo5nSBarBQbaBuaaEAWeQaawac4RWvGAZAMWuAZVpHOeWMrUjefownNmsUTeHwfWqRpefXjnmpSNyzA8aP4CJdR66Hv9otOpMNP0zqMGOrAB6w8AKvf%2BIjm%2B3tV5tv%2BJEN1TbbHsIdWr6FQL9ZR7UKQCe%2F9Q6BxAZ%2BvT5wCXgwgcaVANyS3Vrl3Of%2BDh%2F0EQBKr9g6s1byxq02Q74OlmUVKAWI28FX8ff0oQi54mXc8pG%2BDlZnyllsncnjKYdLBqqjpXhLYpr9TsnIjab3CPJ%2BT7Bu4WGb2a%2B6TJ9nxMib3wxz39wt3VsVs1GH03BsGixNxngNZt45anaBRs5nW9lMg%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230528T213240Z&X-Amz-SignedHeaders=host&X-Amz-Expires=299&X-Amz-Credential=ASIAQ3PHCVTYSFEPE24Z%2F20230528%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=71a2c6dd8a337504f1bbd896cd77909143937c6d64c0ef298c55633fde1a1335&hash=b71d8df859609961d394fcde624c3f213268f3d02913b9f8cce07c220c4be87d&host=68042c943591013ac2b2430a89b270f6af2c76d8dfd086a07176afe7c76c2c61&pii=S030439750100398X&tid=spdf-1c9b4015-4523-447a-9c97-8e3e7130e42a&sid=c55acbbf522bd640052a8c40c79071dc61c7gxrqb&type=client&tsoh=d3d3LnNjaWVuY2VkaXJlY3QuY29t&ua=080) )
* 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)](https://pure.rug.nl/ws/files/78518929/Automated_Planning_for_Ubiquitous_Computing.pdf) )