UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

AUTOMAT BANCAR

Realizatori:
Bogdan Szasz
Bogdan Tamba

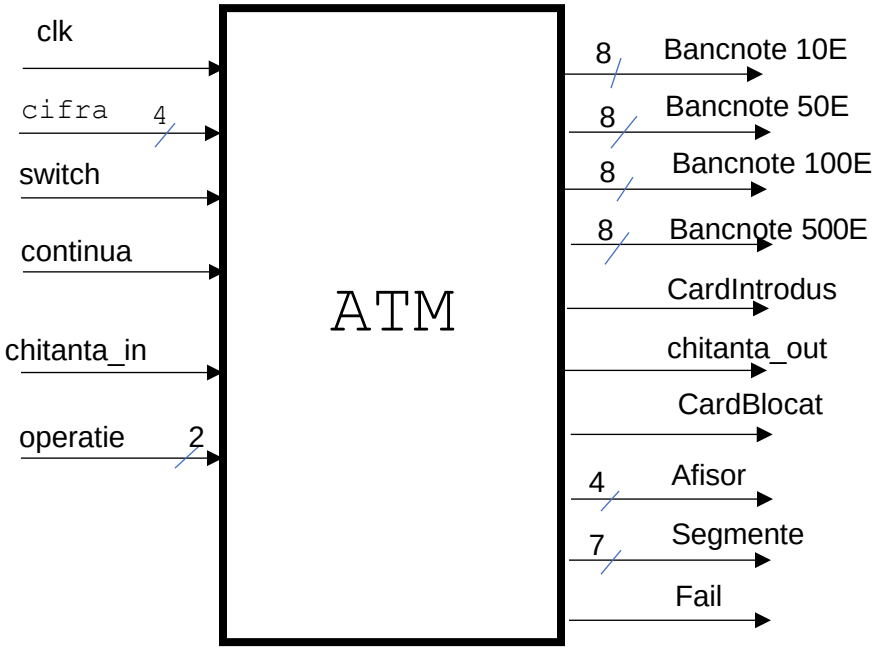
Table of Contents

Specificația Proiectului

Cerința acestui proiect a fost de a implementa un automat bancar ce operează pe sume în EURO. Se presupune că suma maximă care poate fi extrasă este de 1000 EURO. Operații efectuate sunt:

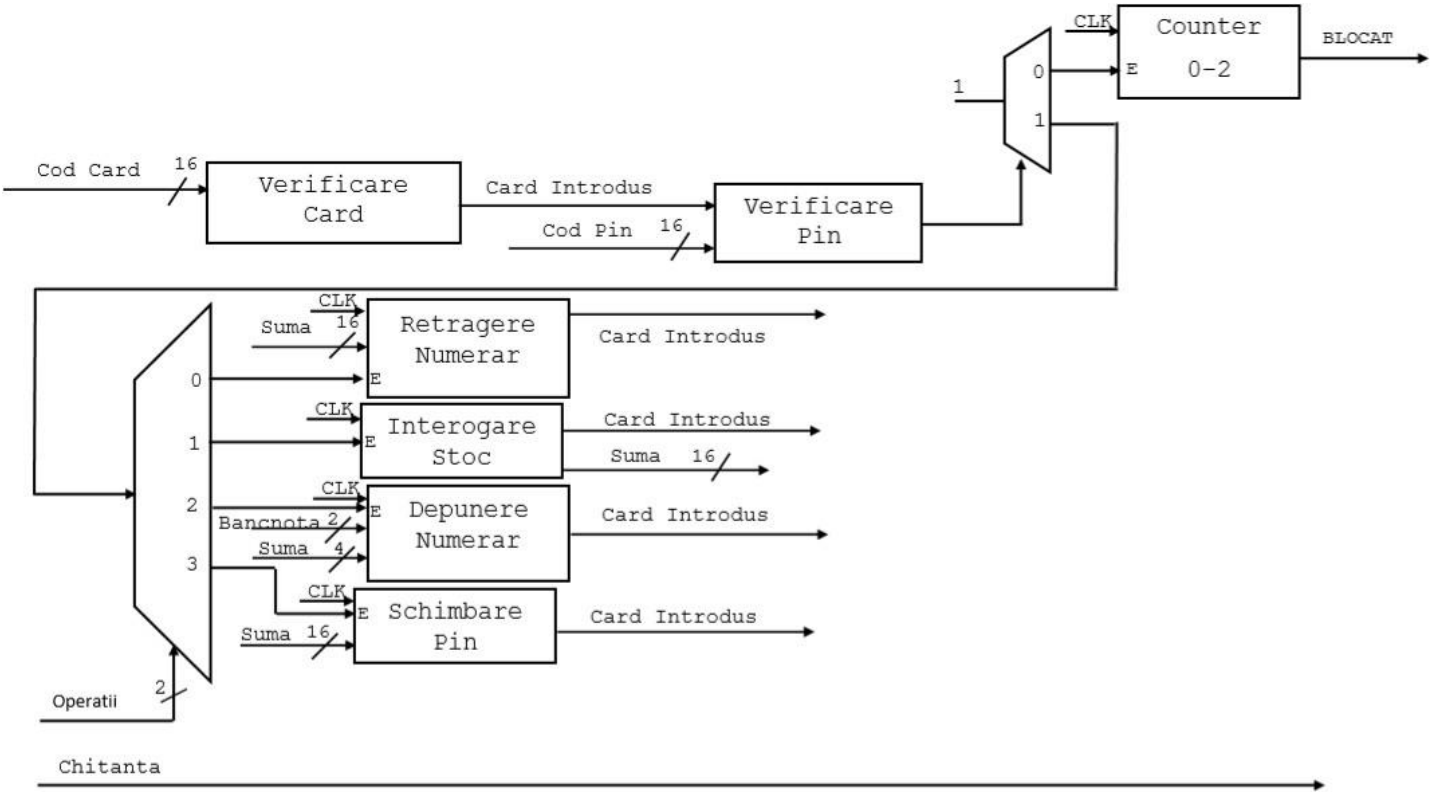
- Retragere numerar
- Depunere numerar
- Schimbare pin
- Interogare sold

Black Box



clk	o	Semnalul de tact specific plăcuței FPGA
cifra	o	Cifra citită
switch	o	Un semnal folosit pentru a comuta între ordinele cifrelor
Continua	o	Un semnal pentru a semnala continuarea parcurgerii
chitanta_in	o	Specifică dacă trebuie sau nu eliberată o chitanță
operatie	o	Tipul operației alese
Bancnote XE	o	Numărul de bancnote de tipul X
CardIntrodus	o	Un semnal ce va avea valoarea ‚1' logic atâta timp cât cardul este introdus în automat
chitanta_out	o	Eliberarea chitanței
CardBlocat	o	Un semnal ce va semnala blocarea cardului după introducerea de 3 ori a unui PIN greșit
Afisor & Segmente	o	Semnale utilizate pentru activarea afișoarelor cu 7 segmente
Fail	o	Va semnala o eroare în cadrul retragerii de numerar

Schema Bloc



Schema bloc are o structură simplă ce urmează următorii pași:

1. Citește și verifica dacă numărul cardului introdus corespunde cu vreunul dintre numerele memorate de automat
2. Dacă s-a găsit cardul, se va citi si se va verifica pinul introdus
3. Introducerea greșită a pinului se va semnala către un numărător. Odată ce valoarea acestuia ajunge la 2 (s-au numărat 3 pinuri greșite), cardul va fi blocat
4. Dacă s-a trecut de verificarea prinului, urmează alegerea operațiilor. Aici sunt 4 posibilități:
    a. Retragerea de numerar – având codul intern „00”
    b. Depunerea de numerar – având codul intern „01”
    c. Schimbarea pinului – având codul intern „10”
    d. Interogarea soldului – având codul intern „11”
5. În funcție de alegerea operației, se va intra într-unul din blocurile specifice.
6. În funcție de dorința clientului, va fi emisa o chitanță

Unitatea de Comandă și Unitatea de Execuție

Unitatea de Execuție a fost prezentată anterior, în cadrul schemei bloc. Aceasta este formată din totalitatea componentelor utilizate de automatul bancar.

Cât despre Unitatea de Comandă, aceasta este reprezentată de fișierul Main.vhd. Acestei unități îi vor fi transmise datele ce urmează a fi prelucrate, iar, în funcție de starea în care se află automatul, acestea vor fi transmise mai departe către componentele specifice unității de execuție.

Unitatea de comandă urmărește 11 stări:

S1: Se vor citi cifrele specifice codului cardului. După citire se va verifica existența cardului în memoria automatului. Dacă s-a citit codul cardului se va trece la S2, dar daca citirea nu a
fost posibilă, se va merge in S1.

S2: În funcție de rezultatul verificării cardului, se va citi și verifca PIN-ul, trecând apoi la S3, sau se va intoarce la citirea codului cardului (S1).

S3: Dacă PIN-ul a fost introdus corect, trece mai departe la S4. Dacă nu, se va contoriza numărul de introduceri greșite, dupăfiecare incrementare întorcându-ne la citirea unui nou PIN (S2). Când acesta va ajunge la 3, cardul va fi blocat în automat și ne vom întoarce la introducerea unui nou card (S1).

S4: Se va introduce tipul operației și, în funcție de aceasta, i se va cere utilizatorului introducerea unui nou număr și se va merge în starea S6 sau se va omite introducerea altor date și va
continua cu S5.

S5: S-a ales interogarea soldului, operația respectivă devineactivă și se merge mai departe la S10.

S6: Operația aleasă necesită introducerea unui alt număr de 4 cifre zecimale. În funcție de aceasta, el semnifică:
`00` -> suma extrasă din bancomat, urmând S7.
`01` -> Numărul de bancnote de 500EURO, 100EURO, 50EURO,respectiv 10EURO, urmând S8.
10 -> noul PIN al cardului, urmând S9.

S7: Se va finaliza citirea și se va activa componenta specifică retragerii apoi se va trece în S10.

S8: Se va finaliza citirea și se va activa componenta specifică depunerii apoi se va trece în S10.

S9: Se va finaliza citirea și se va activa componenta specifică schimbării de PIN apoi se va trece în S10.

S10: Se va oferi chitanța în funcție de dorința clientului și se vor dezactiva toate componentele utilizate anterior. Se va întoarce la starea inițială S1.

S11: Aceasta este o stare de reintoarcere la S1, pentru momentul în
care citirea codului specific cardului nu a fost aprobată. În momentul reîntoarcerii, va fi posibilă citirea.

Felul în care sunt parcurse stările va depinde de 3 semnale: semnalul „continua” modificat direct de catre utilizator și semnalele „st_curenta” și „st_urmatoare” ce reprezintă starea curentă și starea următoare. În momentul în care clientul modifică valoarea semnalului „continua” din 0 în 1, automatul va inițializa starea curentă cu următoarea stare de parcurs.

```
Continuare: process(continua)
begin
    if continua'EVENT and continua = '1' then
        st_curenta <= st_urmatoare;
    end if;
end process Continuare;
```

Lista de Componente

In realizarea proiectului au fost utilizate:

- 3 memorii RAM
- Un multiplexor 1:4
- Un numarator generic pe N biți
- Un comparator generic pe N biți
- Un convertor din BCD in binar
- Un sumator generic pe N biți
- Un scazator generic pe N biți
- Un multiplicator cu 5
- Un multiplicator cu 10
- 4 afișoare cu 7 segmente

Decrierea blocurilor

MemorieClient

MemorieClient este o memorie RAM care memorează, pentru fiecare cod de card al tuturor clienților posibili, câte un ID intern unic. Acest ID este, de fapt, reprezentarea în binar a codului cardului (introdus în BCD). În momentul proiectării, automatul memorează datele a 4 clienți, însă are posibilitatea de a memora toate cele 10000 de combinații posibile pentru codurile cardurilor.

```
entity memorieClient is
    port(
        mode: in std_logic;                                 -- 0 pentru a citi si 1 pentru a scrie
        adr: in std_logic_vector(15 downto 0);              -- adresa
        dataOut: out std_logic_vector(14 downto 0));        -- codul memorat la adresa accesata RAM
end memorieClient;
```

În funcție de modul introdus, ea va citi datele, sau le va scrie. Pentru „mode = 0”, se va citi ID-ul intern aflat la adresa introdusă și va fi returnat către Unitatea de Comandă. Pentru „mode = 0”, se va modifica doar primul bit al ID-ului intern in „1” logic, pentru a semnala blocarea cardului.

MemorieSumaClient

MemorieSumaClient este o altă memorie RAM ce conține sumele din conturile fiecărui client activ al bancomatului. Aceasta primește ca adresă ID-ul intern al cardului și, în funcție de mod, citește suma curentă sau o actualizează cu o valoare nouă.

```
entity memorieSumaClient is
    port(
        clkRAM: in std_logic;                                   -- clock pentru RAM
        mode: in std_logic;                                     -- 0 pentru a citi si 1 pentru a scrie
        adr: in std_logic_vector(14 downto 0);                  -- adresa la care se scrie/citeste
        dataOut: out std_logic_vector(15 downto 0);             -- noua suma de introdus in RAM
        dataIn: in std_logic_vector(15 downto 0));              -- suma memorata la adresa accesata RAM
end memorieSumaClient;
```

MemoriePin

Cea de-a treia memorie, MemoriePin, reprezintă spațiul de stocare al PIN-urilor corespunzătoare cardurilor. La fel ca și MemorieSumaClient, în funcție de modul de funcționare, ea returnează PIN-ul cardului sau îl actualizează cu unul nou.

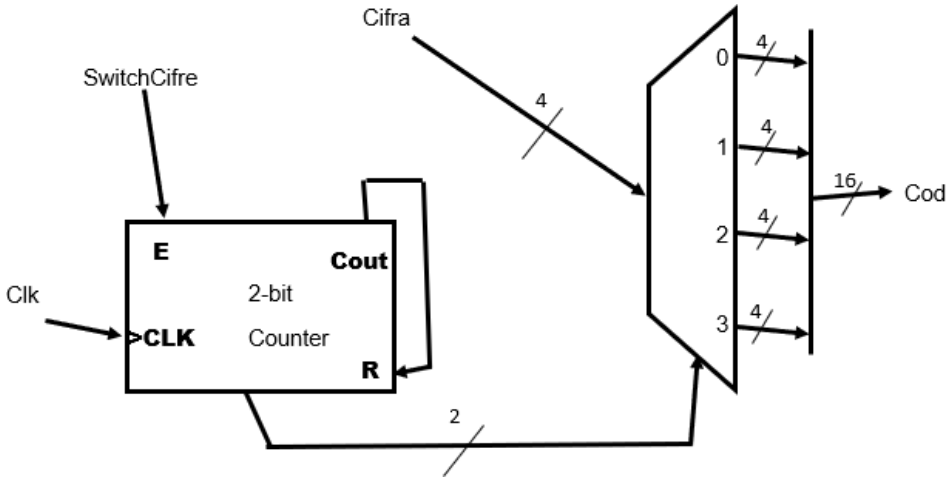
```
entity memoriePin is
    port(
        mode: in std_logic;                                     -- 0 pentru a citi si 1 pentru a scrie
        adr: in std_logic_vector(14 downto 0);                  -- adresa la care se scrie/citeste
        dataIn: in std_logic_vector(15 downto 0);               -- noul pin de introdus in RAM
        dataOut: out std_logic_vector(15 downto 0));            -- pinul memorat la adresa accesata RAM
end memoriePin;
```

Citire

Citire este o componentă utilizată pentru prelucrarea cifrelor introduse de utilizator. Aceasta primește ca input o cifră, un semnal de activare și dezactivare al componentei și un semnal de tact. Semnalul de activare și dezactivarea al componentei va reprezenta trecerea de la un ordin al cifrelor la altul (mii la zeci la sute si la unități). Aceasta trecere va fi făcută cu ajutorul unui numărător cu bucla 0-3.

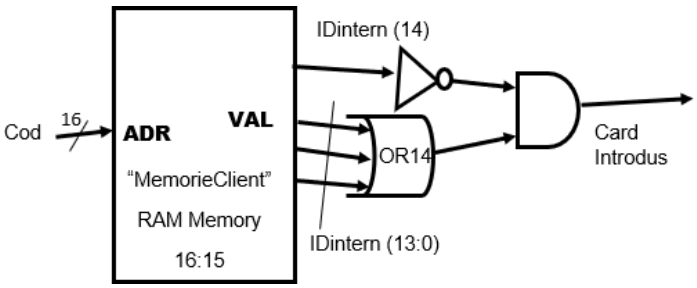
```
begin
L1: COUNTER  generic map(3, 4) port map(clk, "000", '1', switchCifre, '1', TC, ordinCifra);
    process (ordinCifra)
    begin
        if enable = '1' then
        if switchCifre = '1' then                   -- eu am facut mii->sute->zeci->unitati
            if ordinCifra = "001" then              -- se poate modifica si in unitati->zeci->sute->mii
                cod(15 downto 12) <= cifra;         -- mie asa mi s-a parut mai intuitiv
            elsif ordinCifra = "010" then
                cod(11 downto 8) <= cifra;
            elsif ordinCifra = "011" then
                cod(7 downto 4) <= cifra;
            elsif ordinCifra = "100" then
                cod(3 downto 0) <= cifra;
            end if;
        end if;
        end if;
    end process;
```

Pe baza semnalului de tact și al semnalului de activare, numărătorul va fi incrementat iar noua sa stare va reprezenta ordinul cifrei.



Verificare_Card

Aceasta componentă este utilizată pentru a verifica existența cardului în memoria automatului dar și pentru a vedea daca acest card a fost blocat anterior, semn că nu se va mai putea refolosi.



Va fi generat un semnal CardIntrodus ce va avea valoare logică 1 dacă s-a introdus un card valid în ATM, iar daca nu, va avea valoarea logica 0. Acest semnal este generat astfel:

Dacă ne este permisă utlizarea componentei (semnalul enable =‚1') atunci se va face un

```
if enable ='1' then
IDintern_OR := '0';
L1: for i in 0 to 13 loop
    IDintern_OR := IDintern_OR or IDintern(i);
end loop;
CardIntrodus <= not(IDintern(14)) and IDintern_OR;
end if;
```

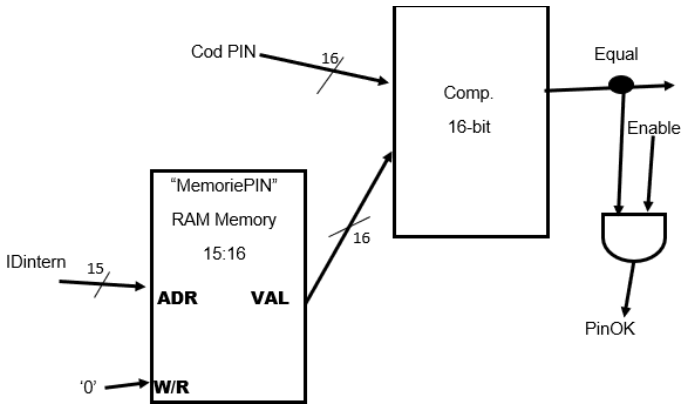
OR intre toți biții ID-ului intern, mai puțin primul bit utilizat pentru semnalarea blocării cardului. Rezultatul va fi adaugat într-o poartă AND alături de al 14-lea bit negat. Acesta este negat deoarece, 0 semnalează un card valid insa „0 and x" va avea mereu valoarea 0.

Verificare_Pin

Aceasta componentă compară pinul introdus de utilizator cu cel aflat în memoria automatului, folosindu-se de un comparator pe 16 biți.

```
M:  memoriePin port map('0', IDintern, "0000000000000000", Pin);
Comp: CompNbit generic map (16) port map(BCD => '1', A => codPin, B => Pin, AequalB => equal );
    pinOK <= equal and enable;
```

Dacă cele două numere sunt egale, se va semnala introducerea unui pin corect prin semnalul „pinOk”.



Interogare_Sold

Aceasta componentă primește ca date de intrare cele 4 cifre ale sumei din contul clientului și le codifică pentru afișarea lor conform regulii:

```
when "0000" => b := "0000001"; --0
when "0001" => b := "1001111"; --1
when "0010" => b := "0010010"; --2
when "0011" => b := "0000110"; --3
when "0100" => b := "1001100"; --4
when "0101" => b := "0100100"; --5
when "0110" => b := "0100000"; --6
when "0111" => b := "0001111"; --7
when "1000" => b := "0000000"; --8
when "1001" => b := "0000100"; --9
when others => b := "1111111"; --Eroare la afisare
```

Depunere_Numerar

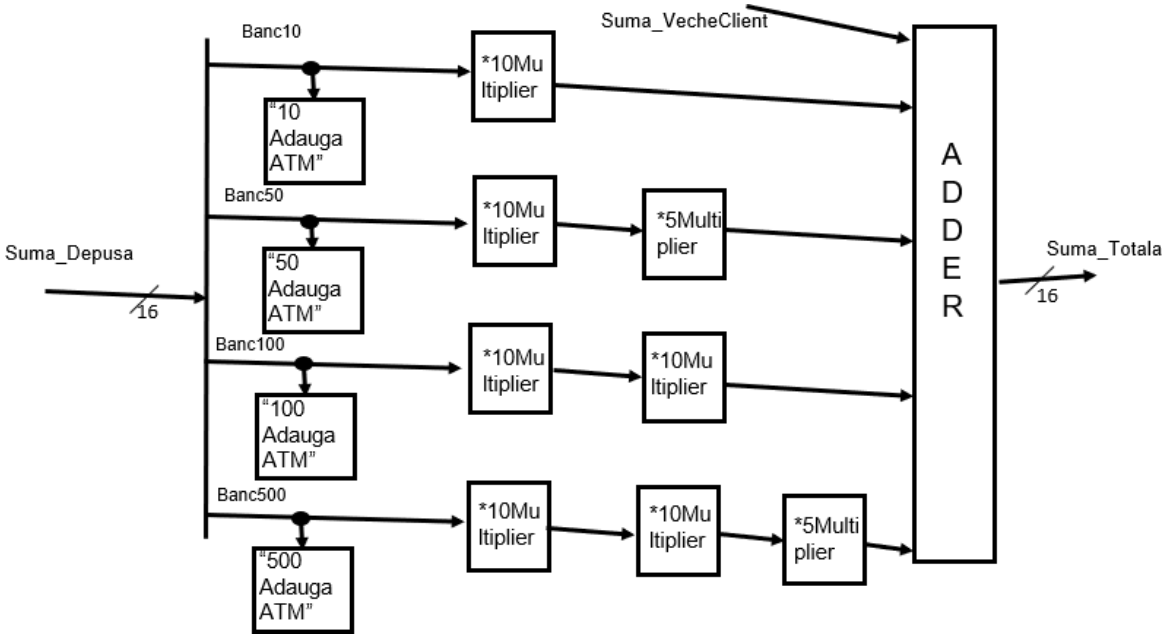
Conform numelui, aceasta componentă se ocupa cu depunerea de numerar în interiorul automatului. Ea va avea ca input:

- Un număr pe 16 biți codificat în BCD cu următoarea semnificație: Primul grup de 4 biți va reprezenta numărul de bancnote de 500EURO introduse, următorul – numărul de bancnote de 100EURO, al treilea
  – numărul de bancnote de 50EURO și ultimul, numărul de bancnotede 10EURO
- 4 semnale pe câte 8 biți reprezentând numărul de bancnote de fiecare tip aflate în automat
- Suma inițiala a clientului

Ca și semnale de ieșire vom avea:

- Noua sumă a clientului

- 4 semnale pe cate 8 biți reprezentând numărul actualizat de bancnote de fiecare tip aflate in automat



Prima dată vom converti numărul de bancnote de 500EURO, 100EURO, etc. în suma pe care o reprezintă, folosind componentele Mull0, Mul5, care realizează înmulțirea cu 10, respectiv 5 a numărului de câte ori e nevoie. Existând o limita a sumei maxime care poate fi depusă odată, dar și o limita a sumei maxime din contul unui client, nu este nevoie să ținem cont de posibilul overflow care va apărea la înmulțirea numerelor. Astfel, vom obține numărul de bancnote înmulțit cu valoarea bancnotei respective, adunându-le pe toate la final, obținând suma totală depusă. Pe aceasta o vom aduna în contul clientului. De asemenea, vom actualiza și numărul de bancnote de fiecare tip din bancomat, adăugând bancnotele depuse.

Retragere_Numerar

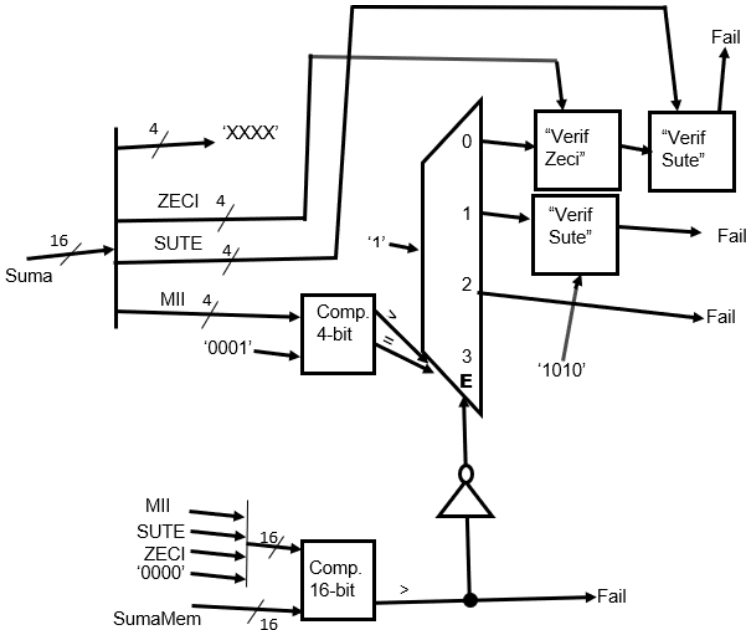
Această componentă va avea ca input:

- Un număr pe 16 biți codificat în BCD reprezentând suma dorită
- 4 semnale pe câte 8 biți reprezentând numărul de bancnote de fiecare tip aflate în automat
- Suma inițială a clientului

Ca și semnale de ieșire vom avea:

- Noua sumă a clientului
- 4 semnale pe cate 8 biți reprezentând numărul actualizat de bancnote de fiecare tip aflate în automat

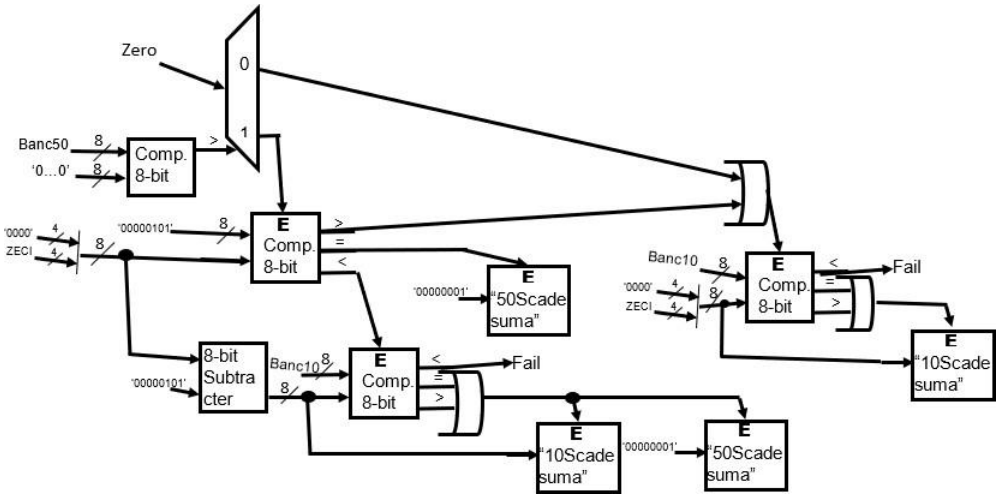
- 4 semnale pe cate 8 biți reprezentând numărul de bancnote de fiecare tip extrase din automat



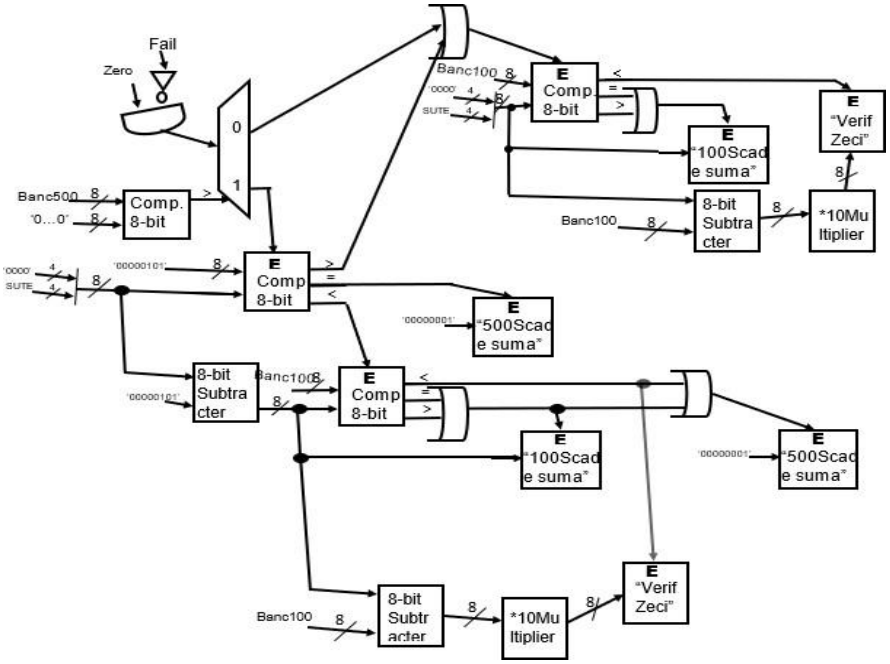
Mai intai, vom verifica daca contul clientului dispune de aceasta suma. Inainte de a scadea suma din cont, vom verifica disponibilitatea bancnotelor ATM-ului.

Numărul fiind dat în BCD, îl vom împarți în zeci, sute, mii (unitățile nu ne interesează). Dacă numărul de mii este mai mare ca 1, sau acesta este egal cu 1 dar numărul de zeci și/sau sute este mai mare decât 0, atunci LED-ul corespunzător mesajului „fail” va fi aprins (ATM-ul nu eliberează mai mult de 1000EURO).

Dacă nu avem de-a face cu miile, vom trece la scăzut de zeci și sute. Mai întâi vom încerca să scădem zecile, dacă reușim trecem mai



departe la sute. Procedeul de scădere al zecilor este unul de tip Greedy, astfel, mai întâi scădem bancnotele de 50, apoi după epuizarea acestora trecem la cele de 10, semnalând ”Fail” în cazul în care nu se va putea. Pentru scăderea sutelor, vom proceda aproximativ la fel, verificând mai întâi bancnotele de 500, apoi 100, apoi trecem la verificare celor de 50 și de 10, din nou.



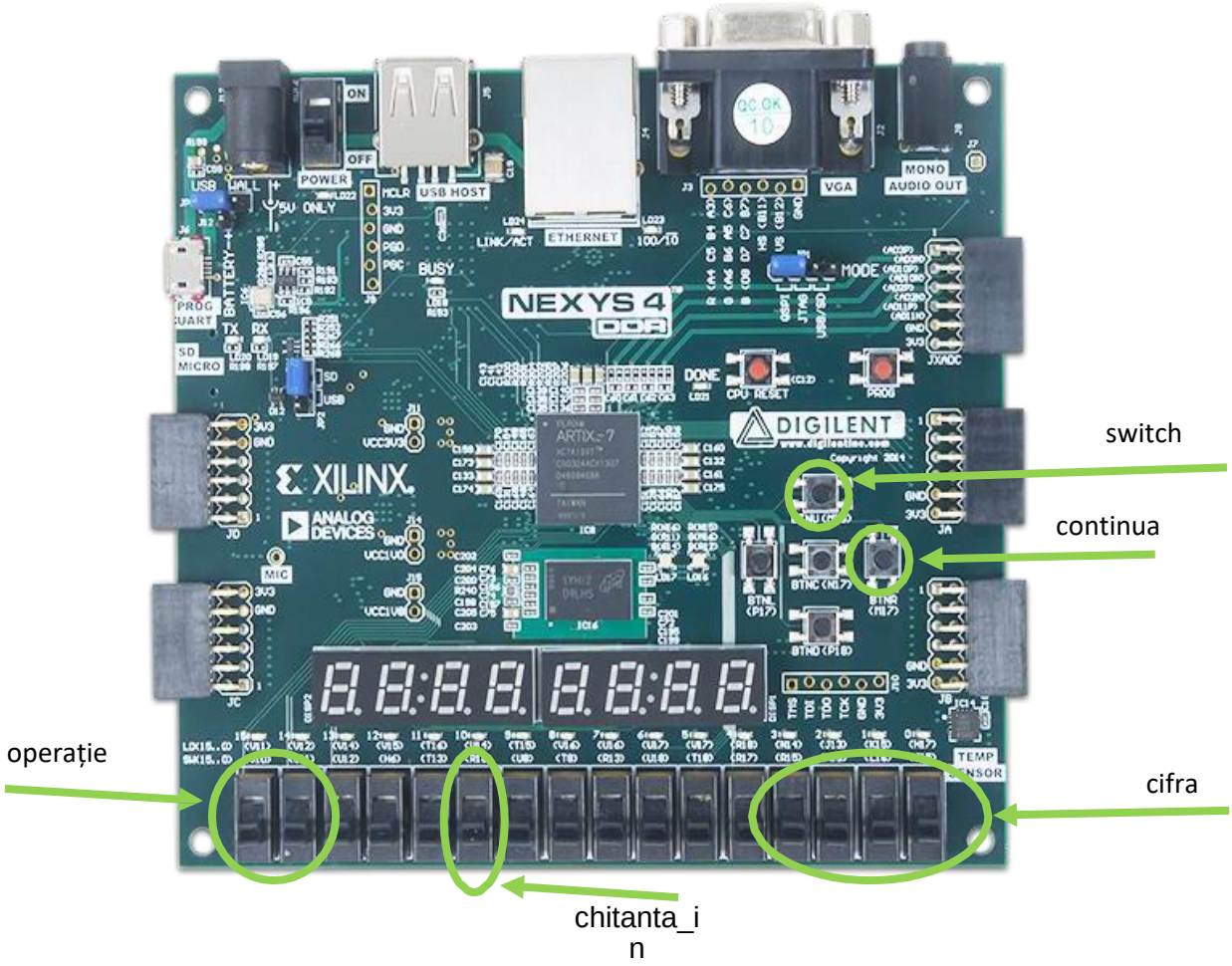
Justificarea Soluției

Am ales soluția de mai sus pentru creearea unui automat bancarcât mai veritabil. Astfel, am introdus și alte operații în afară de retragerea de numerar, precum și posibilitatea blocării cardului odată cu introducerea de 3 ori a unui PIN greșit.

În același timp, am ales memorii pe mai mulți biți decât erau necesari pentru 4 clienți, pentru a oferi posibilitatea adăugării unui număr mai mare de utilizatori.

Am considerat că acest mod de proiectare este un compromis între optimizare și generalizare. În rezolvarea problemei se folosesc multe module secundare, care sunt legate într-o componentă principală. Acest fel de a structura programul ajuta la buna înțelegere a instrucțiunilor utilizate în rezolvarea problemei, fiind favorabil atât proiectantului, cât și utilizatorului acestui proiect.

Instrucțiuni de Uitlizare



Pasul 1: Citirea codului cardului

Pe switch-urile corespunzătoare cifrei se introduce prima cifră, apoi se apasă butonul „switch” pentru memeorarea ei. Acest procedeu se repetă de 4 ori în total, pentru introducerea a 4 cifre.

Pasul 2: Trecem la starea următoare (S11) apăsând „continua”. S11 este o stare de umplutură pentru a nu trece din S1 în S1 și a se forma o buclă infinită. Așa că, apăsând din nou „continuare” se trece în S1, având starea următoare S2. O a treia apăsare a butonului ne va duce în starea S2.

Pasul 3: Citirea PIN-ului

Dacă s-a introdus un card valid, se va repeta procedeul de citire al codului cardului, însă de data aceasta valoare citită va fi corespunzătoare PIN-ului. Dacă nu, apăsarea butonului „continua” ne va intoarce in pasul 1.

Pasul 4: Trecem la starea S3 prin apasarea butonului „continua”. În starea S3, se va verifica dacă s-a introdus un PIN corect, se pregătește starea S4 și se introduce tipul operației. Dacă nu s-a introdus corect numărul pinului și nu am ajuns încă la un total de 3 PIN-uri introduse greșit, se pregătește starea S2. Dacă am introdus de 3 ori greșit, se pregătește starea S1.

Prin apăsarea butonului „continua” se va sări la pasul 5, pasul 3, respectiv pasul 1.

Pasul 5: Dacă operația este cea de interogare sold („11”), se pregătește starea S5. Dacă nu, se va citi un număr de 4 cifre asemenea citirii codului cardului și codului PIN, și se va pregăti starea S6.

Pasul 6: Se apasă „continua”.

În starea S5 se afișează soldul și se specifică dorința unei chitanțe. Se apasă „continua” o dată pentru eliberarea chitanței, și încă o dată pentru a ne reîntoarce la pasul 1.

În starea S6, în funcție de operație se va pregăti una din stările S7, S8, S9. Prin apăsarea butonului „continua” se trece în una din aceste stări și se va specifica dorința unei chitanțe. Încă o apăsare a butonului „continua” ne va duce în starea S10, care printr-o altă apăsare ne va întoarce la pasul 1.

Posibilități de Dezvoltare Ulterioare

Există posibilități de îmbunătățire a acestui proiect ce așteaptă să fie abordate.

Pentru viitor, s-ar putea introduce mai mulți clienți activi, motiv pentru care s-a creat deja o memorie suplimentară.

În același timp, s-ar putea adăuga noi operații precum plata facturilor sau schimbul valutar.

Alte opțiuni de dezvoltare țin de performanța aparatului. Se pot îmbunătății atât memoriile cât și modalitățile de introducere și afisare a datelor.