**Studenti:**

**Coblisan George**

**Corpodean Darius Catalin**

**Grupa: 30215**

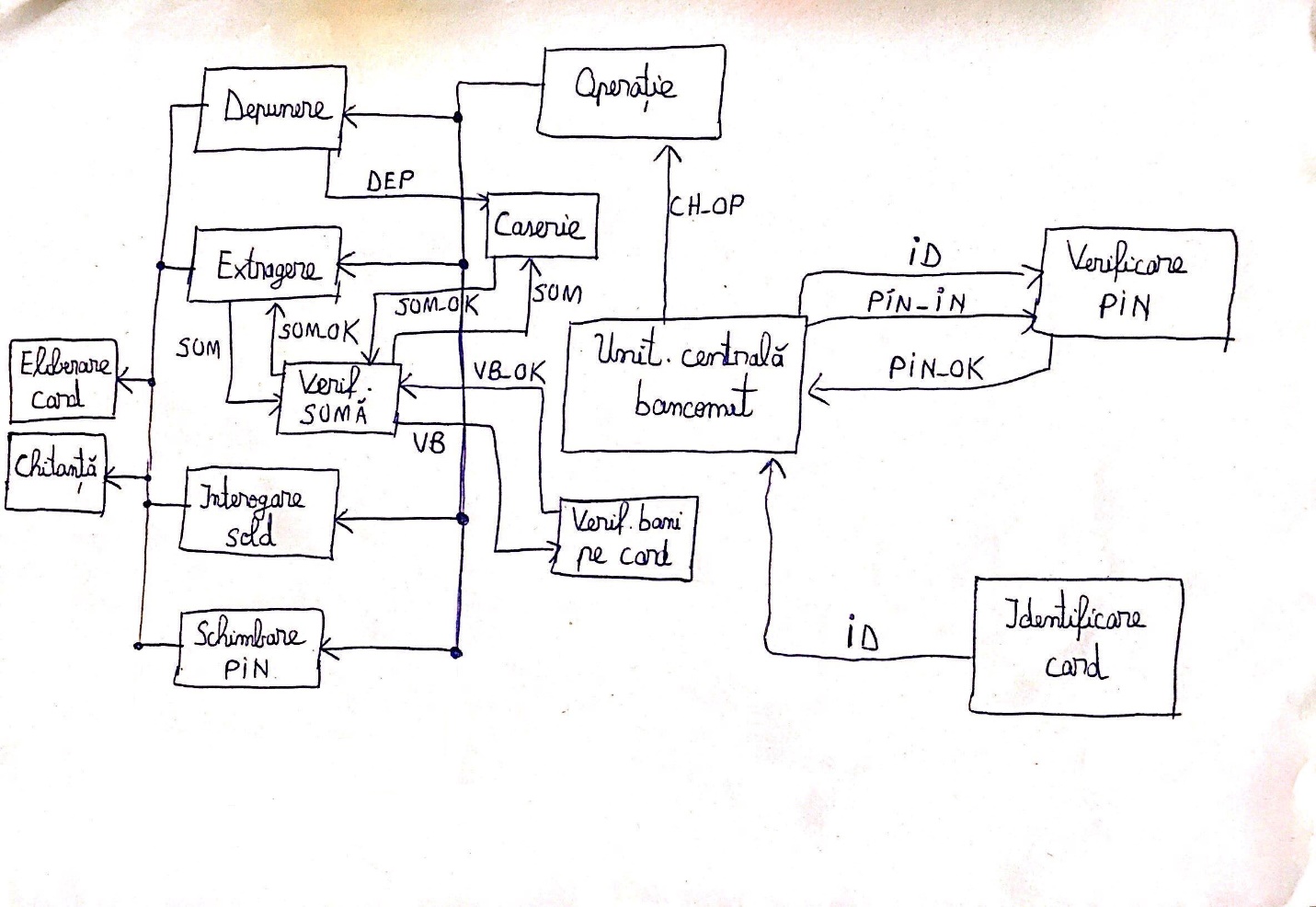
**Automat bancar **

**Cuprins**

1. Specificatie proiect
2. Descriere schema bloc cu componente
3. Proiectare si implementare
4. Lista de componente utilizate
5. Semnificatia notatilor de I/O si a semnalelor interne
6. Justificarea solutiei alese
7. Utilizare si rezultate
8. Posibilitati de dezvoltare ulterioara
9. **Specificatie proiect (capitol realizat de Corpodean Darius)**

Sa se proiecteze un automat bancar de sume in EURO. Se presupune ca suma maxima care poate fi extrasa o data este de maximum 1.000 euro. Initial se efectueaza identificarea cardului si se alege operatia. Vor fi suportate minim 4 carduri/conturi diferite si se vor implementa minimum 4 operatii diferite. Automatul dispune de o casa in care initial se introduce o anumita suma (numar de bancnote de diferite valori). In cazul cererii de eliberare de numerar se introduce suma, se verifica existenta sumei cerute, se vizualizeaza tipurile de bancnote emise si se actualizeaza contul. Apoi se elibereaza cardul, suma si eventual, chitanta.

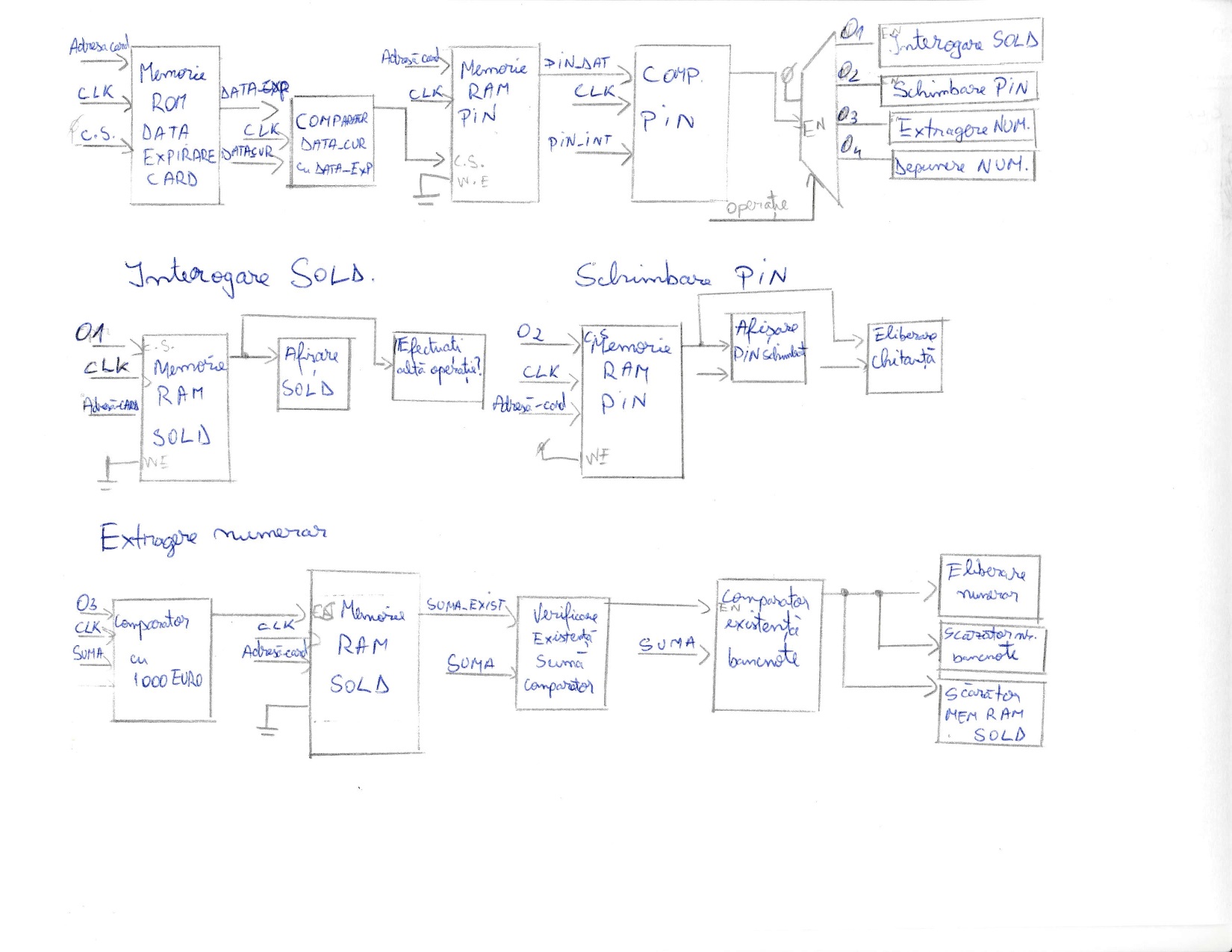
1. **Descriere schema bloc cu componente (capitol realizat de Coblisan George)**

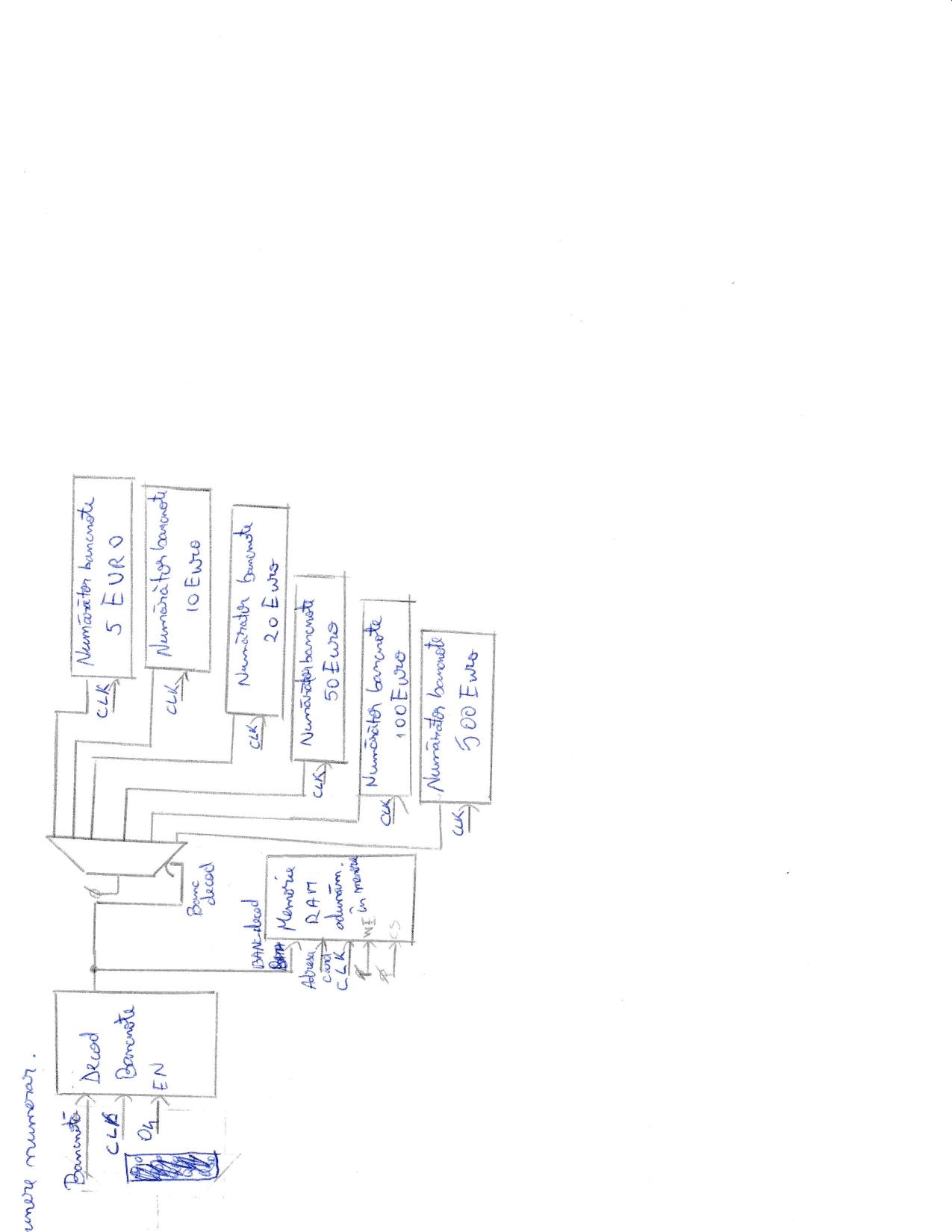


* Unitatea centrala am folosit-o pe desen doar pentru a ne ajuta sa legam primele operatii intre ele, avea mai mult sens, insa mai departe nu a fost nevoie sa o implementam in cod.
* Ordinea efectuarii componentelor este urmatoarea: Identificare card, Verificare PIN, alegerea operatiei (depunere, extragere, interiogare sold, schimbare PIN), eliberare card si eliberare chitanta.
* Schema va fi detaliata si explicata in urmatorul paragraf.

1. **Proiectare si implementare (capitol realizat de amandoi fiind cel mai mare din documentatie)**

Schema de detaliu este urmatoarea:





**Codul pentru componente:**

1. Debouncer (realizat de Corpodean Darius)

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** bistabil\_d **is**

**port(**d**,**clk**,**r**:in** std\_logic**;**

q**:out** std\_logic**);**

**end;**

**architecture** comp **of** bistabil\_d **is**

**begin**

**process(**clk**,**r**)**

**begin**

**if(**r**=**'1'**)** **then**

Q **<=** '0'**;**

**elsif(rising\_edge(**clk**))** **then**

Q **<=** D**;**

**end** **if;**

**end** **process;**

**end;**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** and3 **is**

**port(**a**,**b**,**c**:** **in** std\_logic**;**

y**:out** std\_logic**);**

**end;**

**architecture** flux **of** and3 **is**

**begin**

y**<=** **(**a **and** b**)** **and** c**;**

**end;**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** debouncer **is**

**port(**btn**,** clk**,** r**:in** std\_logic**;**

btn\_db**:out** std\_logic**);**

**end;**

**architecture** struct **of** debouncer **is**

**component** bistabil\_d

**port(**d**,**clk**,**r**:in** std\_logic**;**

q**:out** std\_logic**);**

**end** **component;**

**component** and3

**port(**a**,**b**,**c**:** **in** std\_logic**;**

y**:out** std\_logic**);**

**end** **component** **;**

**signal** s1**,**s2**,**s3**:** std\_logic**;**

**begin**

C1**:** bistabil\_d **port** **map(**Btn**,**clk**,**r**,**s1**);**

C2**:** bistabil\_d **port** **map(**s1**,**clk**,**r**,**s2**);**

C3**:** bistabil\_d **port** **map(**s2**,**clk**,**r**,**s3**);**

C4**:** and3 **port** **map(**s1**,**s2**,**s3**,**btn\_db**);**

**end;**

1. Divizor frecventa (realizat de Corpodean Darius)

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

**entity** divizor **is**

**port(**clk**:** **in** std\_logic**;**

cclk**:** **out** std\_logic**);**

**end** **entity** **;**

**architecture** comp **of** divizor **is**

**signal** clkdiv**:** std\_logic\_vector**(**10 **downto** 0**):=**"00000000000"**;**

**begin**

**process(**clk**)**

**begin**

**if** **(**clk**=**'1'**)** **and** **(**clk'**event)** **then**

clkdiv **<=** clkdiv**+**1**;**

**end** **if;**

**end** **process;**

cclk**<=**clkdiv**(**10**);**

**end** **architecture** **;**

1. Memorie ROM Data Expirare card (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** ROM\_EXP **is**

**port(**cs**:** **in** std\_logic**;**

adresa\_card**:** std\_logic\_vector**(**2 **downto** 0**);**

an\_exp**:out** std\_logic\_vector**(**10 **downto** 0**);**

luna\_exp**:out** std\_logic\_vector**(**3 **downto** 0**));**

**end;**

**architecture** comp **of** ROM\_EXP **is**

**type** an\_memorie **is** **array** **(**0 **to** 4**)** **of** std\_logic\_vector**(**10 **downto** 0**);**

**type** luna\_memorie **is** **array** **(**0 **to** 4**)** **of** std\_logic\_vector**(**3 **downto** 0**);**

**signal** an\_mem**:** an\_memorie**:=(**"11111100100"**,** "11111100101"**,** "11111100110"**,**"11111100101"**,** "11111100011"**);**

**signal** luna\_mem**:** luna\_memorie**:=(**"1100"**,** "1001"**,** "1010"**,** "1001"**,** "1000"**);**

**begin**

**process(**cs**,** adresa\_card**)**

**begin**

**if(**cs**=**'1'**)** **then**

**case** adresa\_card **is**

**when** "000" **=>** an\_exp**<=**an\_mem**(**0**);**

**when** "001" **=>** an\_exp**<=**an\_mem**(**1**);**

**when** "010" **=>** an\_exp**<=**an\_mem**(**2**);**

**when** "011" **=>** an\_exp**<=**an\_mem**(**3**);**

**when** "100" **=>** an\_exp**<=**an\_mem**(**4**);**

**when** **others** **=>** an\_exp**<=**"00000000000"**;**

**end** **case;**

**case** adresa\_card **is**

**when** "000" **=>** luna\_exp**<=**luna\_mem**(**0**);**

**when** "001" **=>** luna\_exp**<=**luna\_mem**(**1**);**

**when** "010" **=>** luna\_exp**<=**luna\_mem**(**2**);**

**when** "011" **=>** luna\_exp**<=**luna\_mem**(**3**);**

**when** "100" **=>** luna\_exp**<=**luna\_mem**(**4**);**

**when** **others** **=>** luna\_exp**<=**"0000"**;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** comp**;**

Memoria ROM pentru expirare card are o entitate cu 3 intrari si 2 iesiri. Sistemul fiind sincron lucreaza doar sub actiunea frontului crescator al clock-ului. Daca chip select e 1 se va cauta adresa cardului care se introduce si se vor extrage luna si anul expirarii cardului.

1. Afisare eroare card (realizat de Coblisan George)

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**STD\_LOGIC\_ARITH**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

**entity** afisare\_eroare\_card **is**

**port(**anod**:** **out** std\_logic\_vector **(**3 **downto** 0**);**

catod**:** **out** std\_logic\_vector **(**6 **downto** 0**);**

clk**:** **in** std\_logic**;**

enable**:** **in** std\_logic**);**

**end** **entity;**

**architecture** eroareCard **of** afisare\_eroare\_card **is**

**signal** cclk**:** std\_logic**;**

**signal** clkdiv**:** std\_logic\_vector**(**10 **downto** 0**)** **:="00000000000";**

**begin**

**process(**clk**)**

**begin**

**if** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

clkdiv **<=** clkdiv**+**1**;**

**end** **if;**

**end** **process;**

cclk**<=**clkdiv**(**10**);**

**process(**cclk**)**

**variable** i**:** integer**:=**3**;**

**begin**

**if** **(**enable**=**'1'**)** **then**

**if(rising\_edge(**cclk**))** **then**

**if(**i**=**3**)** **then**

anod**<=**"0111"**;**

catod**<=**"0110000"**;**

i**:=**2**;**

**elsif(**i**=**2**)** **then**

anod**<=**"1011"**;**

catod**<=**"1111010"**;**

i**:=**1**;**

**elsif(**i**=**1**)** **then**

anod**<=**"1101"**;**

catod**<=**"1111010"**;**

i**:=**0**;**

**else**

anod**<=**"1110"**;**

catod**<=**"0100100"**;**

i**:=**3**;**

**end** **if;**

**end** **if;**

**else**

anod**<=**"ZZZZ"**;**

catod**<=**"ZZZZZZZ"**;**

**end** **if;**

**end** **process;**

**end** eroareCard**;**

1. Comparator data curenta cu data expirarii (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** COMP\_DATA **is**

**port(**an\_exp**:**std\_logic\_vector**(**10 **downto** 0**);**

luna\_exp**:in** std\_logic\_vector**(**3 **downto** 0**);**

data\_ok**:** **out** std\_logic**);**

**end;**

**architecture** comp **of** COMP\_DATA **is**

**signal** luna\_curenta**:** std\_logic\_vector**(**3 **downto** 0**):=** "0010"**;**

**signal** an\_curent**:** std\_logic\_vector**(**10 **downto** 0**):=** "11111100100"**;**

**begin**

**process(**an\_exp**)**

**begin**

**if(**an\_exp**=**"00000000000"**)** **then** data\_ok**<=**'0'**;**

**elsif(**an\_exp**<**an\_curent**)** **then** data\_ok**<=**'0'**;**

**elsif(**an\_exp**>**an\_curent**)** **then** data\_ok**<=**'1'**;**

**elsif(**luna\_exp**>**luna\_curenta**)** **then** data\_ok**<=**'1'**;**

**else** data\_ok**<=**'0'**;**

**end** **if;**

**end** **process;**

**end** comp**;**

Comparator primeste anul expirarii si luna expirarii cu luna si anul curent si in cazul in care cardul nu este expirat se transmite pe iesire 1 ca se activeze urmatoarea componenta.

1. Memorie RAM PIN (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** RAM\_PIN **is**

**port(**cs**,**we**,**clk**:** **in** std\_logic**;**

adresa\_card**:in** std\_logic\_vector**(**2 **downto** 0**);**

pin\_de\_schimbat**:in** std\_logic\_vector**(**15 **downto** 0**);**

pin\_dat**:out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **entity** **;**

**architecture** comp **of** RAM\_PIN **is**

**type** PIN\_MEMORIE **is** **array** **(**0 **to** 3**)** **of** STD\_LOGIC\_VECTOR**(**15 **downto** 0**);**

**signal** PIN\_MEM**:** PIN\_MEMORIE**:=(**"0010010011111011"**,** "0000101001011110"**,** "0001001100000000"**,** "0001011111101101"**);**

**begin**

**process(**clk**)**

**begin**

**if(**clk**=**'1'**)** and **(**clk'**event)then**

**if(**cs**=**'1'**)** and **(**we**=**'0'**)** **then**

**case** adresa\_card **is**

**when** "000" **=>** PIN\_dat**<=**PIN\_MEM**(**0**);**

**when** "001" **=>** PIN\_dat**<=**PIN\_MEM**(**1**);**

**when** "010" **=>** PIN\_dat**<=**PIN\_MEM**(**2**);**

**when** "011" **=>** PIN\_dat**<=**PIN\_MEM**(**3**);**

**when** **others** **=>** PIN\_dat**<=**"ZZZZZZZZZZZZZZZZ"**;**

**end** **case;**

**elsif(**cs**=**'1'**)** and **(**we**=**'1'**)** **then**

**case** adresa\_card **is**

**when** "000" **=>** PIN\_MEM**(**0**)<=**pin\_de\_schimbat**;**

**when** "001" **=>** PIN\_MEM**(**1**)<=**pin\_de\_schimbat**;**

**when** "010" **=>** PIN\_MEM**(**2**)<=**pin\_de\_schimbat**;**

**when** "011" **=>** PIN\_MEM**(**3**)<=**pin\_de\_schimbat**;**

**when** **others** **=>** PIN\_MEM**(**0**)<=**PIN\_MEM**(**0**);**

**end** **case;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** comp**;**

In memoria RAM pentru PIN se tin minte toate pin-urile cardurilor introduse, si la primirea adresei cardului folosit se va elibera pin-ul corespunzator pentru a fi ulterior comparat. Cand write enable este activat se va rescrie PINUL cu PINUL care este pe intrarea de pin schimbat.

1. Selectare PIN (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Selectare\_PIN\_SUME **is**

**port(**clk**,**en**,**data**:in** std\_logic**;**

pin**,**pin\_nou**,**suma**:** **out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** Selectare\_PIN\_SUME **is**

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

**begin**

**process(**clk**)**

**variable** op**:** std\_logic\_vector**(**1 **downto** 0**):=**"00"**;**

**begin**

**if(**clk**=**'1' **and** clk'**event)** **then**

**if(**en**=**'1'**)** **then**

op**:=**op**+**1**;**

**end** **if;**

**end** **if;**

operatie**<=**op**;**

**end** **process;**

pin**<=**data **when** **(**operatie**=**"00" **)** **else** '0'**;**

pin\_nou**<=**data **when** **(**operatie**=**"01" **)** **else** '0'**;**

suma**<=**data **when** **(**operatie**=**"10" **)** **else** '0'**;**

**end** comp**;**

Aceasta componenta este alcatuita dintr-un numarator combinat cu un demultiplexor. Se numara cu ajutorul unui clock operatia pe care dorim sa o efectuam: daca nu se numara, pinul va primi selectia de date din memorie, daca se numara o data pinul nou va primi selectia de date din memorie si daca se numara de doua ori suma de pe card va primi selectia de date din memorie.

1. Numarator PIN unitati ( realizat de Coblisan George)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Pin\_unit **is**

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **entity;**

**architecture** comp **of** pin\_unit **is**

**signal** num**:** integer**:=**0**;**

**begin**

**process** **(**clk**)**

**begin**

**if(**clk**=**'1' **and** clk'**event)** **then**

**if(**en**=**'1'**)** **then**

num**<=** num **+** 1**;**

**end** **if;**

**end** **if;**

unit**<=**num**;**

**end** **process;**

**end** **architecture;**

1. Numarator PIN zeci (realizat de Coblisan George)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Pin\_zeci **is**

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **entity;**

**architecture** comp **of** pin\_zeci **is**

**signal** num**:** integer**:=**0**;**

**begin**

**process** **(**clk**)**

**begin**

**if(**clk**=**'1' **and** clk'**event)** **then**

**if(**en**=**'1'**)** **then**

num**<=** num **+** 1**;**

**end** **if;**

**end** **if;**

unit**<=**num**;**

**end** **process;**

**end** **architecture** **;**

1. Numarator PIN sute (realizat de Coblisan George)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Pin\_sute **is**

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **entity;**

**architecture** comp **of** pin\_sute **is**

**signal** num**:** integer**:=**0**;**

**begin**

**process** **(**clk**)**

**begin**

**if(**clk**=**'1' **and** clk'**event)** **then**

**if(**en**=**'1'**)** **then**

num**<=** num **+** 1**;**

**end** **if;**

**end** **if;**

unit**<=**num**;**

**end** **process;**

**end** **architecture** **;**

1. Numarator PIN mii (realizat de Coblisan George)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Pin\_mii **is**

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **entity;**

**architecture** comp **of** pin\_mii **is**

**signal** num**:** integer**:=**0**;**

**begin**

**process** **(**clk**)**

**begin**

**if(**clk**=**'1' **and** clk'**event)** **then**

**if(**en**=**'1'**)** **then**

num**<=** num **+** 1**;**

**end** **if;**

**end** **if;**

unit**<=**num**;**

**end** **process;**

**end** **architecture** **;**

1. Asamblare PIN (realizat de Coblisan George)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**use** IEEE**.**std\_logic\_arith**.all;**

**entity** asamblare\_pin **is**

**port(**u**,**z**,**s**,**m**:** **in** integer**;**

en**:** **in** std\_logic**;**

pin**:** **out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** asamblare\_pin **is**

**signal** nr**:**integer**;**

**begin**

**process(**u**,**z**,**s**,**m**)**

**variable** nr**:**integer**;**

**begin**

**if(**en**=**'1'**)** **then**

nr**:=**u**+**z**\***10**+**s**\***100**+**m**\***1000**;**

pin**<=**conv\_std\_logic\_vector**(**nr**,** 16**);**

**end** **if;**

**end** **process;**

**end** **architecture** **;**

Cu ajutorul acestei componente se prelucreaza cifrele pinului pe care le stim datorita numaratoarelor anterioare si se pun toate cifrele la un loc.

1. Comparator PIN (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** COMP\_PIN **is**

**port(**pin\_memorie**,** pin\_intrare**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

pin\_ok**:out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** COMP\_PIN **is**

**begin**

**process(**pin\_memorie**,** pin\_intrare**)**

**begin**

**if(**pin\_memorie**=**pin\_intrare**)** **then** pin\_ok**<=**'1'**;**

**else** pin\_ok**<=**'0'**;**

**end** **if;**

**end** **process;**

**end** comp**;**

Comparatorul de pin verfica daca pin-ul introdus este identic cu cel al cardului si in caz afirmativ transmite pe iesirea PIN\_OK semnalul 1 pentru a putea fi activata componenta urmatoare(DEMUX OPERATII).

1. Afisare Eroare PIN (realizat de Coblisan George)

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

**entity** afisare\_eroare\_pin **is**

**port(**anod**:** **out** std\_logic\_vector **(**3 **downto** 0**);**

catod**:** **out** std\_logic\_vector **(**6 **downto** 0**);**

clk**:** **in** std\_logic**;**

enable**:** **in** std\_logic**);**

**end** **entity;**

**architecture** eroarePin **of** afisare\_eroare\_pin **is**

**signal** cclk**:** std\_logic**;**

**signal** clkdiv**:** std\_logic\_vector**(**10 **downto** 0**)** **:="00000000000";**

**begin**

**process(**clk**)**

**begin**

**if** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

clkdiv **<=** clkdiv**+**1**;**

**end** **if;**

**end** **process;**

cclk**<=**clkdiv**(**10**);**

**process(**cclk**)**

**variable** i**:** integer**:=**3**;**

**begin**

**if** **(**enable**=**'1'**)** **then**

**if(rising\_edge(**cclk**))** **then**

**if(**i**=**3**)** **then**

anod**<=**"0111"**;**

catod**<=**"0110000"**;**

i**:=**2**;**

**elsif(**i**=**2**)** **then**

anod**<=**"1011"**;**

catod**<=**"1111010"**;**

i**:=**1**;**

**elsif(**i**=**1**)** **then**

anod**<=**"1101"**;**

catod**<=**"1111010"**;**

i**:=**0**;**

**else**

anod**<=**"1110"**;**

catod**<=**"0011000"**;**

i**:=**3**;**

**end** **if;**

**end** **if;**

**else**

anod**<=**"ZZZZ"**;**

catod**<=**"ZZZZZZZ"**;**

**end** **if;**

**end** **process;**

**end** eroarePin**;**

1. Demultiplexor Operatii (realizat de Corpodean Darius)

**entity** DMUX\_OP **is**

**port(**en**,**data**:in** bit**;**

operatie**:in** bit\_vector**(**1 **downto** 0**);**

o1**,**o2**,**o3**,**o4**:** **out** bit**);**

**end** **entity;**

**architecture** comp **of** DMUX\_OP **is**

**begin**

**process(**en, operatie, data**)**

**begin**

**if(**en**=**'1'**)** **then**

**case** operatie **is**

**when** "00" **=>** o1**<=**data**;**

**when** "01" **=>** o2**<=**data**;**

**when** "10" **=>** o3**<=**data**;**

**when** "11" **=>** o4**<=**data**;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** comp**;**

Demultiplexorul pentru operatii va pune pe iesirea dorita pentru operatie:

* Interogare SOLD(O1)
* Schimbare PIN(O2)
* Extragere Numerar(O3)
* Depunere NUMERAR(O4)
* Ramificand astfel urmatorul punct prin selectarea operatiunii dorite si continuarea pe acea ramura.

1. Decodificator mii (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**entity** conv\_test **is**

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** conv\_test**;**

**architecture** Behavioral **of** conv\_test **is**

**begin**

b **<=** **to\_integer(**unsigned**(**a**));**

**end** Behavioral**;**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**use** IEEE**.**std\_logic\_arith**.all** **;**

**entity** decodificator\_mii **is**

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** decodificator\_mii **is**

**component** conv\_test

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** **component;**

**signal** b**,**c**:** integer**;**

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

**begin**

c1**:** conv\_test **port** **map(**suma**,** b**);**

**process(**suma**,** b**)**

**variable** c**:** integer**;**

**begin**

c**:=(**b**/**1000**);**

c**:=(**c **mod** 10**);**

d**<=** conv\_std\_logic\_vector**(**c**,** 4**);**

**end** **process;**

**process(**d**)**

**begin**

**case** d **is**

**when** "0000" **=>** nr\_decodificat**<=**"0000001"**;**

**when** "0001" **=>** nr\_decodificat**<=**"1001111"**;**

**when** "0010" **=>** nr\_decodificat**<=**"0010010"**;**

**when** "0011" **=>** nr\_decodificat**<=**"0000110"**;**

**when** "0100" **=>** nr\_decodificat**<=**"1001100"**;**

**when** "0101" **=>** nr\_decodificat**<=**"0100100"**;**

**when** "0110" **=>** nr\_decodificat**<=**"0100000"**;**

**when** "0111" **=>** nr\_decodificat**<=**"0001111"**;**

**when** "1000" **=>** nr\_decodificat**<=**"0000000"**;**

**when** "1001" **=>** nr\_decodificat**<=**"0000100"**;**

**when** **others** **=>** **null;**

**end** **case;**

**end** **process;**

**end;**

1. Decodificator sute (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**entity** conv\_test **is**

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** conv\_test**;**

**architecture** Behavioral **of** conv\_test **is**

**begin**

b **<=** **to\_integer(**unsigned**(**a**));**

**end** Behavioral**;**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**use** IEEE**.**std\_logic\_arith**.all** **;**

**entity** decodificator\_sute **is**

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** decodificator\_sute **is**

**component** conv\_test

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** **component;**

**signal** b**,**c**:** integer**;**

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

**begin**

c1**:** conv\_test **port** **map(**suma**,** b**);**

**process(**suma**,** b**)**

**variable** c**:** integer**;**

**begin**

c**:=(**b**/**100**);**

c**:=(**c **mod** 10**);**

d**<=** conv\_std\_logic\_vector**(**c**,** 4**);**

**end** **process;**

**process(**d**)**

**begin**

**case** d **is**

**when** "0000" **=>** nr\_decodificat**<=**"0000001"**;**

**when** "0001" **=>** nr\_decodificat**<=**"1001111"**;**

**when** "0010" **=>** nr\_decodificat**<=**"0010010"**;**

**when** "0011" **=>** nr\_decodificat**<=**"0000110"**;**

**when** "0100" **=>** nr\_decodificat**<=**"1001100"**;**

**when** "0101" **=>** nr\_decodificat**<=**"0100100"**;**

**when** "0110" **=>** nr\_decodificat**<=**"0100000"**;**

**when** "0111" **=>** nr\_decodificat**<=**"0001111"**;**

**when** "1000" **=>** nr\_decodificat**<=**"0000000"**;**

**when** "1001" **=>** nr\_decodificat**<=**"0000100"**;**

**when** **others** **=>** **null;**

**end** **case;**

**end** **process;**

**end;**

1. Decodificator zeci (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**entity** conv\_test **is**

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** conv\_test**;**

**architecture** Behavioral **of** conv\_test **is**

**begin**

b **<=** **to\_integer(**unsigned**(**a**));**

**end** Behavioral**;**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**use** IEEE**.**std\_logic\_arith**.all** **;**

**entity** decodificator\_sute **is**

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** decodificator\_sute **is**

**component** conv\_test

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** **component;**

**signal** b**,**c**:** integer**;**

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

**begin**

c1**:** conv\_test **port** **map(**suma**,** b**);**

**process(**suma**,** b**)**

**variable** c**:** integer**;**

**begin**

c**:=(**b**/**100**);**

c**:=(**c **mod** 10**);**

d**<=** conv\_std\_logic\_vector**(**c**,** 4**);**

**end** **process;**

**process(**d**)**

**begin**

**case** d **is**

**when** "0000" **=>** nr\_decodificat**<=**"0000001"**;**

**when** "0001" **=>** nr\_decodificat**<=**"1001111"**;**

**when** "0010" **=>** nr\_decodificat**<=**"0010010"**;**

**when** "0011" **=>** nr\_decodificat**<=**"0000110"**;**

**when** "0100" **=>** nr\_decodificat**<=**"1001100"**;**

**when** "0101" **=>** nr\_decodificat**<=**"0100100"**;**

**when** "0110" **=>** nr\_decodificat**<=**"0100000"**;**

**when** "0111" **=>** nr\_decodificat**<=**"0001111"**;**

**when** "1000" **=>** nr\_decodificat**<=**"0000000"**;**

**when** "1001" **=>** nr\_decodificat**<=**"0000100"**;**

**when** **others** **=>** **null;**

**end** **case;**

**end** **process;**

**end;**

1. Decodificator unitati (realizat de Corpodean Darius)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**entity** conv\_test **is**

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** conv\_test**;**

**architecture** Behavioral **of** conv\_test **is**

**begin**

b **<=** **to\_integer(**unsigned**(**a**));**

**end** Behavioral**;**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**USE** ieee**.**numeric\_std**.ALL;**

**use** IEEE**.**std\_logic\_arith**.all** **;**

**entity** decodificator\_unit **is**

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** decodificator\_unit **is**

**component** conv\_test

**Port** **(** a **:** **in** STD\_LOGIC\_VECTOR **(**15 **downto** 0**);**

b **:** **out** integer**);**

**end** **component;**

**signal** b**,**c**:** integer**;**

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

**begin**

c1**:** conv\_test **port** **map(**suma**,** b**);**

**process(**suma**,** b**)**

**variable** c**:** integer**;**

**begin**

c**:=(**b **mod** 10**);**

d**<=** conv\_std\_logic\_vector**(**c**,** 4**);**

**end** **process;**

**process(**d**)**

**begin**

**case** d **is**

**when** "0000" **=>** nr\_decodificat**<=**"0000001"**;**

**when** "0001" **=>** nr\_decodificat**<=**"1001111"**;**

**when** "0010" **=>** nr\_decodificat**<=**"0010010"**;**

**when** "0011" **=>** nr\_decodificat**<=**"0000110"**;**

**when** "0100" **=>** nr\_decodificat**<=**"1001100"**;**

**when** "0101" **=>** nr\_decodificat**<=**"0100100"**;**

**when** "0110" **=>** nr\_decodificat**<=**"0100000"**;**

**when** "0111" **=>** nr\_decodificat**<=**"0001111"**;**

**when** "1000" **=>** nr\_decodificat**<=**"0000000"**;**

**when** "1001" **=>** nr\_decodificat**<=**"0000100"**;**

**when** **others** **=>** **null;**

**end** **case;**

**end** **process;**

**end;**

1. Memorie RAM SOLD (realizat de Coblisan George)

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** RAM\_SOLD **is**

**port(**cs**,**we**,**clk**:** **in** std\_logic**;**

adresa\_card**:** **in** std\_logic\_vector**(**2 **downto** 0**);**

suma\_noua**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

suma\_exist**:** **out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** RAM\_SOLD **is**

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

**signal** SOLD\_MEM**:** SOLD\_MEMORIE**:=(**"0001100100000000"**,** "0001001110001000"**,** "0001111101000000"**,** "0010010100011100"**);**

**begin**

**process(**clk**)**

**begin**

**if(**clk**=**'1' and clk'**event)** **then**

**if(**cs**=**'1' and we**=**'0'**)** **then**

**case** adresa\_card **is**

**when** "000" **=>** suma\_exist**<=**SOLD\_MEM**(**0**);**

**when** "001" **=>** suma\_exist**<=**SOLD\_MEM**(**1**);**

**when** "010" **=>** suma\_exist**<=**SOLD\_MEM**(**2**);**

**when** "011" **=>** suma\_exist**<=**SOLD\_MEM**(**3**);**

**when** **others** **=>** suma\_exist**<=**"0000000000000000"**;**

**end** **case;**

**elsif(**cs**=**'1' and we**=**'1'**)** **then**

**case** adresa\_card **is**

**when** "000" **=>** SOLD\_MEM**(**0**)<=**suma\_noua**;**

**when** "001" **=>** SOLD\_MEM**(**1**)<=**suma\_noua**;**

**when** "010" **=>** SOLD\_MEM**(**2**)<=**suma\_noua**;**

**when** "011" **=>** SOLD\_MEM**(**3**)<=**suma\_noua**;**

**when** **others** **=>** SOLD\_MEM**(**0**)<=**SOLD\_MEM**(**0**);**

**end** **case;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** comp**;**

In memoria RAM pentru sold se tin minte toate sumele introduse si la primirea adresei cardului folosit se va elibera suma noua folosita pentru viitoarele operatii. Cand write enable este activat se va rescrie soldul cu soldul care este pe intrarea de sold schimbat.

1. Afisare PIN schimbat (realizat de Coblisan George)

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

**entity** afisare\_pin\_schimbat **is**

**port(**anod**:** **out** std\_logic\_vector **(**3 **downto** 0**);**

catod**:** **out** std\_logic\_vector **(**6 **downto** 0**);**

clk**:** **in** std\_logic**;**

enable**:** **in** std\_logic**);**

**end** **entity;**

**architecture** arh **of** afisare\_pin\_schimbat **is**

**signal** cclk**:** std\_logic**;**

**signal** clkdiv**:** std\_logic\_vector**(**10 **downto** 0**)** **:="00000000000";**

**begin**

**process(**clk**)**

**begin**

**if** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

clkdiv **<=** clkdiv**+**1**;**

**end** **if;**

**end** **process;**

cclk**<=**clkdiv**(**10**);**

**process(**cclk**)**

**variable** i**:** integer**:=**3**;**

**begin**

**if** **(**enable**=**'1'**)** **then**

**if(rising\_edge(**cclk**))** **then**

**if(**i**=**3**)** **then**

anod**<=**"0111"**;**

catod**<=**"0110000"**;**

i**:=**2**;**

**elsif(**i**=**2**)** **then**

anod**<=**"1011"**;**

catod**<=**"1111010"**;**

i**:=**1**;**

**elsif(**i**=**1**)** **then**

anod**<=**"1101"**;**

catod**<=**"1111010"**;**

i**:=**0**;**

**else**

anod**<=**"1110"**;**

catod**<=**"0011000"**;**

i**:=**3**;**

**end** **if;**

eliberare\_chitanta<='1';

**end** **if;**

**else**

anod**<=**"ZZZZ"**;**

catod**<=**"ZZZZZZZ"**;**

**end** **if;**

**end** **process;**

**end** arh**;**

1. Afisare SOLD (realizat de Coblisan George)

**library** IEEE**;**

**use** IEEE**.**STD\_LOGIC\_1164**.ALL;**

**use** IEEE**.**STD\_LOGIC\_ARITH**.ALL;**

**use** IEEE**.**STD\_LOGIC\_UNSIGNED**.ALL;**

**entity** afisare\_sold **is**

**port(**anod**:** **out** std\_logic\_vector **(**3 **downto** 0**);**

catod**:** **out** std\_logic\_vector **(**6 **downto** 0**);**

dec\_zeci**:** **in** std\_logic\_vector **(**6 **downto** 0**);**

dec\_sute**:** **in** std\_logic\_vector **(**6 **downto** 0**);**

dec\_mii**:** **in** std\_logic\_vector **(**6 **downto** 0**);**

clk**:** **in** std\_logic**;**

enable**:** **in** std\_logic**);**

**end** **entity;**

**architecture** arh **of** afisare\_sold **is**

**signal** cclk**:** std\_logic**;**

**signal** clkdiv**:** std\_logic\_vector**(**10 **downto** 0**)** **:="00000000000";**

**begin**

**process(**clk**)**

**begin**

**if** **(**clk**=**'1'**)** **and** **(**clk'**event)** **then**

clkdiv **<=** clkdiv**+**1**;**

**end** **if;**

**end** **process;**

cclk**<=**clkdiv**(**10**);**

**process(**cclk**)**

**variable** i**:** integer**:=**3**;**

**begin**

**if** **(**enable**=**'1'**)** **then**

**if(rising\_edge(**cclk**))** **then**

**if(**i**=**3**)** **then**

anod**<=**"0111"**;**

catod**<=**"1001000"**;**

i**:=**2**;**

**elsif(**i**=**2**)** **then**

anod**<=**"1011"**;**

catod**<=**dec\_mii**;**

i**:=**1**;**

**elsif(**i**=**1**)** **then**

anod**<=**"1101"**;**

catod**<=**dec\_sute**;**

i**:=**0**;**

**else**

anod**<=**"1110"**;**

catod**<=**dec\_zeci**;**

i**:=**3**;**

**end** **if;**

**end** **if;**

**else**

anod**<=**"ZZZZ"**;**

catod**<=**"ZZZZZZZ"**;**

**end** **if;**

**end** **process;**

**end** arh**;**

1. Comparator 1000 EURO (realizat de Coblisan George)

**library** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** COMP\_1000 **is**

**port(**suma\_ceruta**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

o3**:** **in** std\_logic**;**

suma\_ok**:** **out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** COMP\_1000 **is**

**begin**

**process(**o3**,** suma\_ceruta**)**

**begin**

**if(**o3**=**'1'**)** **then**

**if(**suma\_ceruta**<**"001111101000"**)** **then** suma\_ok**<=**'1'**;**

**else** suma\_ok**<=**'0'**;**

**end** **if;**

**end** **if;**

**end** **process;**

**end** comp

Comparatorul cu 1000 de EURO verifica daca suma dorita pentru extragere este mai mica decat suma maxima de extragere 1000 de EURO si daca este se va activa componenta urmatoare care va lucra cu urmatorii pasi necesari extragerii.

1. Comparator cu SUME (realizat de Coblisan George)

**library** ieee**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** COMP\_SUMA **is**

**port(**suma\_exist**,** suma\_ceruta**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

suma\_ok**:** **out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** COMP\_SUMA **is**

**begin**

**process(**suma\_exist**,** suma\_ceruta**)**

**begin**

**if(**suma\_ceruta**<**suma\_exist**)** **then** suma\_ok**<=**'1'**;**

**else** suma\_ok**<=**'0'**;**

**end** **if;**

**end** **process;**

**end** comp**;**

Comparatorul de sume verifica daca suma ceruta pentru extragere exista in contul aferent cardului si daca da se va verifica in continuare daca exista bani in aparat.

1. Comparator existenta bancnote (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**numeric\_std**.all;**

**entity** Comp\_BANC **is**

**port(**en**:in** std\_logic**;**

suma**:in** std\_logic\_vector**(**15 **downto** 0**);**

n5**,**n10**,**n20**,**n50**,**n100**,**n500**:** **in** std\_logic\_vector**(**7 **downto** 0**);**

d5**,**d10**,**d20**,**d50**,**d100**,**d500 **:out** std\_logic\_vector**(**7 **downto** 0**);**

banc\_ok**:out** std\_logic

**)** **;**

**end** **entity;**

**architecture** comp **of** Comp\_BANC **is**

**type** val\_banc **is** **array** **(**0 **to** 5**)** **of** std\_logic\_vector**(**15 **downto** 0**);**

**signal** banc\_plus**:** val\_banc**:=(**"0000000000000101"**,** "0000000000001010"**,** "0000000000010100"**,** "0000000000110010"**,** "0000000001100100"**,** "0000000111110100"**);**

**begin**

**process(**en**,** suma**,**n5**,**n10**,**n20**,**n50**,**n100**,**n500 **)**

**variable** x**:**std\_logic\_vector**(**15 **downto** 0**):=**"0000000000000000"**;**

**variable** ok**:**std\_logic**;**

**variable** b1**,**b2**,**b3**,**b4**,**b5**,**b6**:** std\_logic\_vector**(**7 **downto** 0**);**

**variable** s**:**std\_logic\_vector**(**7 **downto** 0**):=**"00000001"**;**

**constant** max\_iterations**:**natural**:=**100**;**

**begin**

ok**:=**'1'**;**

b1**:=**n5**;**

b2**:=**n10**;**

b3**:=**n20**;**

b4**:=**n50**;**

b5**:=**n100**;**

b6**:=**n500**;**

**if(**en**=**'1'**)** **then**

L1**:for** i **in** 1 **to** max\_iterations **loop**

**if(**suma**/=**x**)** **then**

**exit** L1 **when** ok**=**'0'**;**

**if(**b6**/=**"00000000"**)** **and** **(**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**5**)))<=**std\_logic\_vector**(**unsigned**(**suma**)))** **then**

x**:=**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**5**)));**

b6**:=**std\_logic\_vector**(**unsigned**(**b6**)-**unsigned**(**s**));**

**elsif(**b5**/=**"00000000"**)** **and** **(**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**4**)))<=**std\_logic\_vector**(**unsigned**(**suma**)))** **then**

x**:=**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**4**)));**

b5**:=**std\_logic\_vector**(**unsigned**(**b5**)-**unsigned**(**s**));**

**elsif(**b4**/=**"00000000"**)** **and** **(**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**3**)))<=**std\_logic\_vector**(**unsigned**(**suma**)))** **then**

x**:=**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**3**)));**

b4**:=**std\_logic\_vector**(**unsigned**(**b4**)-**unsigned**(**s**));**

**elsif(**b3**/=**"00000000"**)** **and** **(**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**2**)))<=**std\_logic\_vector**(**unsigned**(**suma**)))** **then**

x**:=**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**2**)));**

b3**:=**std\_logic\_vector**(**unsigned**(**b3**)-**unsigned**(**s**));**

**elsif(**b2**/=**"00000000"**)** **and** **(**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**1**)))<=**std\_logic\_vector**(**unsigned**(**suma**)))** **then**

x**:=**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**1**)));**

b2**:=**std\_logic\_vector**(**unsigned**(**b2**)-**unsigned**(**s**));**

**elsif(**b1**/=**"00000000"**)** **and** **(**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**0**)))<=**std\_logic\_vector**(**unsigned**(**suma**)))** **then**

x**:=**std\_logic\_vector**(**unsigned**(**x**)+**unsigned**(**banc\_plus**(**0**)));**

b1**:=**std\_logic\_vector**(**unsigned**(**b1**)-**unsigned**(**s**));**

**else** ok**:=**'0'**;**

**end** **if;**

**end** **if;**

**end** **loop;**

**else** ok**:=**'U'**;**

**end** **if;**

banc\_ok**<=**ok**;**

d5**<=**b1**;**

d10**<=**b2**;**

d20**<=**b3**;**

d50**<=**b4**;**

d100**<=**b5**;**

d500**<=**b6**;**

**end** **process;**

**end** **architecture;**

Comparatorul de bancnote primeste intreg numarul de bancnote de fiecare valoare si verifica daca se poate obtine suma ceruta cu nr. de bancnote din aparat. Se verifica intr-un loop fiecare nr de bancnote daca se poate aduna astfel incat sa se poata extrage suma. In caz afirmativ, se vor scadea nr de bancote, si se va active iesirea banc\_ok, urmand ca in numaratoare nr de bancnote noi sa fie introdus in numaratoare, si SOLD-UL sa fie scazut.

1. Scazator Memorie RAM SOLD (realizat de Coblisan George)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

-- ieee.std\_logic\_arith.all;

**use** ieee**.**numeric\_std**.all;**

**entity** scazator\_sold **is**

**port** **(** en**:** **in** std\_logic**;**

suma\_extrasa**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

sold\_existent**:in** std\_logic\_vector**(**15 **downto** 0**);**

sold\_nou**:** **out** std\_logic\_vector**(**15 **downto** 0**);**

scadere\_ok**:** **out** std\_logic**;**

eliberare\_numerar**:** **out** std\_logic**;**

eliberare\_card**:** **out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** scazator\_sold **is**

**type** val\_banc **is** **array** **(**0 **to** 5**)** **of** std\_logic\_vector**(**15 **downto** 0**);**

**signal** banc\_plus**:** val\_banc**:=(**"0000000000000101"**,** "0000000000001010"**,** "0000000000010100"**,** "0000000000110010"**,** "0000000001100100"**,** "0000000111110100"**);**

**begin**

**process(**en**,**suma\_extrasa**,** sold\_existent**)**

**begin**

**if** en**=**'1' **then**

sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_existent**)-**unsigned**(**suma\_extrasa**));**

scadere\_ok**<=**'1'**;**

eliberare\_card**<=**'1'**;**

eliberare\_numerar**<=**'1'**;**

**end** **if;**

**end** **process;**

**end** comp**;**

Scade din soldul existent suma extrasa, urmand ca noul sold sa fie pastrat in memoria RAM specifica.

1. Sumator Memorie RAM (realizat de Coblisan George)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

-- ieee.std\_logic\_arith.all;

**use** ieee**.**numeric\_std**.all;**

**entity** Sumator\_Sold **is**

**port** **(**decod\_ok**:** **in** std\_logic**;**

banc\_decod**:** **in** std\_logic\_vector**(**2 **downto** 0**);**

sold\_vechi**:in** std\_logic\_vector**(**15 **downto** 0**);**

sold\_nou**:** **out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Sumator\_Sold **is**

**type** val\_banc **is** **array** **(**0 **to** 5**)** **of** std\_logic\_vector**(**15 **downto** 0**);**

**signal** banc\_plus**:** val\_banc**:=(**"0000000000000101"**,** "0000000000001010"**,** "0000000000010100"**,** "0000000000110010"**,** "0000000001100100"**,** "0000000111110100"**);**

**begin**

**process(**decod\_ok**,** banc\_decod**,** sold\_vechi**)**

**begin**

**if** decod\_ok**=**'0' **then**

**case** banc\_decod **is**

**when** "000" **=>** sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_vechi**)+**unsigned**(**banc\_plus**(**0**)));**

**when** "001" **=>** sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_vechi**)+**unsigned**(**banc\_plus**(**1**)));**

**when** "010" **=>** sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_vechi**)+**unsigned**(**banc\_plus**(**2**)));**

**when** "011" **=>** sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_vechi**)+**unsigned**(**banc\_plus**(**3**)));**

**when** "100" **=>** sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_vechi**)+**unsigned**(**banc\_plus**(**4**)));**

**when** "101" **=>** sold\_nou**<=** std\_logic\_vector**(**unsigned**(**sold\_vechi**)+**unsigned**(**banc\_plus**(**5**)));**

**when** **others=>** **null;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** comp**;**

Aduna la sold-ul existent pe rand bancnotele de 5, 10, 20, 50, 100 si respectiv 500 in modul depunere numerar .

1. Decodificator Bancnote (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** Decod\_Banc **is**

**port(**o4**:in** std\_logic**;**

bancnota**:** **in** std\_logic\_vector**(**8 **downto** 0**);**

banc\_decod**:** **out** std\_logic\_vector**(**2 **downto** 0**);**

banc\_ok**:** **out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** Decod\_Banc **is**

**begin**

**process(**o4**,** bancnota**)**

**begin**

**if(**o4**=**'1'**)** **then**

**case** bancnota **is**

**when** "000000101" **=>** banc\_decod**<=**"000"**;**

**when** "000001010" **=>** banc\_decod**<=**"001"**;**

**when** "000010100" **=>** banc\_decod**<=**"010"**;**

**when** "000110010" **=>** banc\_decod**<=**"011"**;**

**when** "001100100" **=>** banc\_decod**<=**"100"**;**

**when** "111110100" **=>** banc\_decod**<=**"101"**;**

**when** **others=>** **null;**

**end** **case;**

**banc\_ok<=’1’;**

**end** **if;**

**end** **process;**

**end** comp**;**

Se decodifica bancontele pentru a fi mai usor sa se lucreze in continuare. In loc sa folosim un demultiplexor 1:512 pentru a folosi doar 6 intrari din 512, decodificam bancnotele si ne vom folosi de un DEMUX 1:8 unde vom folosi 6 din 8 iesiri.

1. Demultiplexor Bancnote (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** DMUX\_BANC **is**

**port(**data**,**en**:in** std\_logic**;**

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

n1**,**n2**,**n3**,**n4**,**n5**,**n6**:** **out** std\_logic**);**

**end** **entity;**

**architecture** comp **of** DMUX\_BANC **is**

**begin**

**process(**data**,** en**,** sel**)**

**begin**

**if(**en**=**'1'**)** **then**

**case** sel **is**

**when** "000" **=>** n1**<=**data**;**

**when** "001" **=>** n2**<=**data**;**

**when** "010" **=>** n3**<=**data**;**

**when** "011" **=>** n4**<=**data**;**

**when** "100" **=>** n5**<=**data**;**

**when** "101" **=>** n6**<=**data**;**

**when** **others=>** **null;**

**end** **case;**

**end** **if;**

**end** **process;**

**end** comp**;**

In functie de bancnota decodificata se activeaza numaratorul necesar pentru a Numara bancnotele care exista in bancomat.

1. Numarator 5 euro (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Num\_5 **is**

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Num\_5 **is**

**signal** num**:** std\_logic\_vector**(**7 **downto** 0**):=**"01000000" **;**

**begin**

**process(**clk**,** clr**)**

**begin**

**if(**clr**=**'1'**)** **then** num**<=**"00000000"**;**

**elsif** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

**if(**en**=**'1'**)** **then**

**if(**pl**=**'1'**)then**

num**<=**Data**;**

**else**

num**<=**num**+**1**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Q**<=**num**;**

**end** comp**;**

1. Numarator 10 EURO (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Num\_10 **is**

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Num\_10 **is**

**signal** num**:** std\_logic\_vector**(**7 **downto** 0**):=**"01000000" **;**

**begin**

**process(**clk**,** clr**)**

**begin**

**if(**clr**=**'1'**)** **then** num**<=**"00000000"**;**

**elsif** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

**if(**en**=**'1'**)** **then**

**if(**pl**=**'1'**)then**

num**<=**Data**;**

**else**

num**<=**num**+**1**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Q**<=**num**;**

**end** comp**;**

1. Numarator 20 EURO (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Num\_20 **is**

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Num\_20 **is**

**signal** num**:** std\_logic\_vector**(**7 **downto** 0**):=**"01000000" **;**

**begin**

**process(**clk**,** clr**)**

**begin**

**if(**clr**=**'1'**)** **then** num**<=**"00000000"**;**

**elsif** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

**if(**en**=**'1'**)** **then**

**if(**pl**=**'1'**)then**

num**<=**Data**;**

**else**

num**<=**num**+**1**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Q**<=**num**;**

**end** comp**;**

1. Numarator 50 EURO (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Num\_50 **is**

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Num\_50 **is**

**signal** num**:** std\_logic\_vector**(**7 **downto** 0**):=**"01000000" **;**

**begin**

**process(**clk**,** clr**)**

**begin**

**if(**clr**=**'1'**)** **then** num**<=**"00000000"**;**

**elsif** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

**if(**en**=**'1'**)** **then**

**if(**pl**=**'1'**)then**

num**<=**Data**;**

**else**

num**<=**num**+**1**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Q**<=**num**;**

**end** comp**;**

1. Numarator 100 EURO (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Num\_100 **is**

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Num\_100 **is**

**signal** num**:** std\_logic\_vector**(**7 **downto** 0**):=**"01000000" **;**

**begin**

**process(**clk**,** clr**)**

**begin**

**if(**clr**=**'1'**)** **then** num**<=**"00000000"**;**

**elsif** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

**if(**pl**=**'1'**)then**

**if(**en**=**'1'**)** **then**

num**<=**Data**;**

**else**

num**<=**num**+**1**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Q**<=**num**;**

**end** comp**;**

1. Numarator 500 EURO (realizat de Corpodean Darius)

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**use** ieee**.**std\_logic\_unsigned**.all;**

**entity** Num\_500 **is**

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** comp **of** Num\_500 **is**

**signal** num**:** std\_logic\_vector**(**7 **downto** 0**):=**"01000000" **;**

**begin**

**process(**clk**,** clr**)**

**begin**

**if(**clr**=**'1'**)** **then** num**<=**"00000000"**;**

**elsif** **(**clk**=**'1'**)** and **(**clk'**event)** **then**

**if(**pl**=**'1'**)then**

**if(**en**=**'1'**)** **then**

num**<=**Data**;**

**else**

num**<=**num**+**1**;**

**end** **if;**

**end** **if;**

**end** **if;**

**end** **process;**

Q**<=**num**;**

**end** comp**;**

**Codul pentru ansamblu:**

(realizat de amandoi)

**library** work**;**

**library** IEEE**;**

**USE** ieee**.**std\_logic\_1164**.ALL;**

**entity** ATM\_BANCAR **is**

**port(**clk**,** clk1**,** clk2**,** clk3**,** clk4**,** clk5**:in** std\_logic**;**

adresa\_card**:** **in** std\_logic\_vector**(**2 **downto** 0**);**

op**:** **in** std\_logic\_vector**(**1 **downto** 0**);**

bancnota\_introdusa**:** **in** std\_logic\_vector**(**8 **downto** 0**);**

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

catod**:** **out** std\_logic\_vector**(**6 **downto** 0**);**

eliberare\_card**:** **out** std\_logic**;**

eliberare\_chitanta**:** **out** std\_logic**;**

eliberare\_numerar**:** **out** std\_logic**);**

**end** **entity;**

**architecture** STRUCTURALA **of** ATM\_BANCAR **is**

**component** ROM\_EXP

**port(** cs**:** **in** std\_logic**;**

adresa\_card**:** **in** std\_logic\_vector**(**2 **downto** 0**);**

an\_exp**:out** std\_logic\_vector**(**10 **downto** 0**);**

luna\_exp**:out** std\_logic\_vector**(**3 **downto** 0**));**

**end** **component;**

**component** COMP\_DATA

**port(**

an\_exp**:in** std\_logic\_vector**(**10 **downto** 0**);**

luna\_exp**:in** std\_logic\_vector**(**3 **downto** 0**);**

data\_ok**:** **out** std\_logic**);**

**end** **component;**

**component** RAM\_PIN

**port(**cs**,**we**,**clk**:** **in** std\_logic**;**

adresa\_card**:in** std\_logic\_vector**(**2 **downto** 0**);**

pin\_de\_schimbat**:in** std\_logic\_vector**(**15 **downto** 0**);**

pin\_dat**:out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **component;**

**component** COMP\_PIN

**port(**

pin\_memorie**,** pin\_intrare**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

pin\_ok**:out** std\_logic **);**

**end** **component;**

**component** DMUX\_OP

**port(**en**,**data**:in** std\_logic**;**

operatie**:in** std\_logic\_vector**(**1 **downto** 0**);**

o1**,**o2**,**o3**,**o4**:** **out** std\_logic**);**

**end** **component;**

**component** RAM\_SOLD

**port(**cs**,**we**,**clk**:** **in** std\_logic**;**

adresa\_card**:** **in** std\_logic\_vector**(**2 **downto** 0**);**

suma\_noua**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

suma\_exist**:** **out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **component;**

**component** COMP\_1000

**port(**

o3**:in** std\_logic**;**

suma\_ceruta**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

suma\_ok**:** **out** std\_logic**);**

**end** **component;**

**component** COMP\_SUMA

**port(**

suma\_exist**,** suma\_ceruta**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

suma\_ok**:** **out** std\_logic**);**

**end** **component;**

**component** afisare\_sold

**port(**anod**:** **out** std\_logic\_vector **(**3 **downto** 0**);**

catod**:** **out** std\_logic\_vector **(**6 **downto** 0**);**

dec\_zeci**:** **in** std\_logic\_vector **(**6 **downto** 0**);**

dec\_sute**:** **in** std\_logic\_vector **(**6 **downto** 0**);**

dec\_mii**:** **in** std\_logic\_vector **(**6 **downto** 0**);**

clk**:** **in** std\_logic**;**

enable**:** **in** std\_logic**);**

**end** **component;**

**component** afisare\_pin\_schimbat

**port(**anod**:** **out** std\_logic\_vector **(**3 **downto** 0**);**

catod**:** **out** std\_logic\_vector **(**6 **downto** 0**);**

clk**:** **in** std\_logic**;**

enable**:** **in** std\_logic**;**

eliberare\_chitanta**:** **out** std\_logic**);**

**end** **component;**

**component** Comp\_BANC

**port(**en**:in** std\_logic**;**

suma**:in** std\_logic\_vector**(**15 **downto** 0**);**

n5**,**n10**,**n20**,**n50**,**n100**,**n500**:** **in** std\_logic\_vector**(**7 **downto** 0**);**

d5**,**d10**,**d20**,**d50**,**d100**,**d500 **:out** std\_logic\_vector**(**7 **downto** 0**);**

banc\_ok**:out** std\_logic

**)** **;**

**end** **component;**

**component** Num\_5

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **component;**

**component** Num\_10

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **component;**

**component** Num\_20

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **component;**

**component** Num\_50

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **component;**

**component** Num\_100

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **component;**

**component** Num\_500

**port(**clk**,** en**,** clr**,** pl**:** **in** std\_logic**;**

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

Q**:** **out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **component;**

**component** scazator\_sold

**port** **(** en**:** **in** std\_logic**;**

suma\_extrasa**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

sold\_existent**:in** std\_logic\_vector**(**15 **downto** 0**);**

sold\_nou**:** **out** std\_logic\_vector**(**15 **downto** 0**);**

scadere\_ok**:** **out** std\_logic**;**

eliberare\_numerar**:** **out** std\_logic**;**

eliberare\_card**:** **out** std\_logic**);**

**end** **component;**

**component** Decod\_Banc

**port(**o4**:in** std\_logic**;**

bancnota**:** **in** std\_logic\_vector**(**8 **downto** 0**);**

banc\_decod**:** **out** std\_logic\_vector**(**2 **downto** 0**);**

banc\_ok**:** **out** std\_logic**);**

**end** **component;**

**component** Sumator\_Sold

**port** **(**

banc\_decod**:** **in** std\_logic\_vector**(**2 **downto** 0**);**

sold\_vechi**:in** std\_logic\_vector**(**15 **downto** 0**);**

sold\_nou**:** **out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **component;**

**component** DMUX\_BANC

**port(**data**,**en**:in** std\_logic**;**

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

n1**,**n2**,**n3**,**n4**,**n5**,**n6**:** **out** std\_logic**);**

**end** **component;**

**component** decodificator\_mii

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **component;**

**component** decodificator\_sute

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **component;**

**component** decodificator\_zeci

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **component;**

**component** decodificator\_unit

**port(**suma**:** **in** std\_logic\_vector**(**15 **downto** 0**);**

nr\_decodificat**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **component;**

**component** Pin\_mii

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **component;**

**component** Pin\_sute

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **component;**

**component** Pin\_zeci

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **component;**

**component** Pin\_unit

**port(**clk**,** en**:** **in** std\_logic**;**

unit**:** **out** integer**);**

**end** **component;**

**component** asamblare\_pin

**port(**u**,**z**,**s**,**m**:** **in** integer**;**

en**:** **in** std\_logic**;**

pin**:** **out** std\_logic\_vector**(**15 **downto** 0**));**

**end** **component;**

**component** Selectare\_pin\_sume

**port(**clk**,** en**,** data**:in** std\_logic**;**

pin**,**pin\_nou**,**suma**:** **out** std\_logic**);**

**end** **component;**

**component** debouncer

**port(**btn**,** clk**,** r**:in** std\_logic**;**

btn\_db**:out** std\_logic**);**

**end** **component;**

**component** divizor

**port(**clk**:** **in** std\_logic**;**

cclk**:** **out** std\_logic**);**

**end** **component** **;**

**signal** an\_exp**:** std\_logic\_vector**(**10 **downto** 0**);**

**signal** luna\_exp**:** std\_logic\_vector**(**3 **downto** 0**);**

**signal** data\_ok**,** suma\_ok**,** suma\_da**,** banc\_ok**,** scad\_ok**,**bancnota\_ok**,** we**:** std\_logic**;**

**signal** pin\_dat**,** pin\_nou1**:** std\_logic\_vector**(**15 **downto** 0**);**

**signal** pin\_ok**:** std\_logic**;**

**signal** o1**,** o2**,** o3**,** o4**:** std\_logic**;**

**signal** suma\_exist**,** suma\_noua**,** sold\_nou**,** suma\_existenta**,** suma\_exist1**,** suma\_exist2**,** sold\_exist1**:** std\_logic\_vector**(**15 **downto** 0**);**

**signal** dec\_unit**,** dec\_zeci**,** dec\_sute**,** dec\_mii**:** std\_logic\_vector**(**6 **downto** 0**);**

**signal** n5**,** n10**,** n20**,** n50**,** n100**,** n500**,** d5**,** d10**,** d20**,** d50**,** d100**,** d500**:** std\_logic\_vector**(**7 **downto** 0**);**

**signal** banc\_decod**:** std\_logic\_vector**(**2 **downto** 0**);**

**signal** nr1**,** nr2**,** nr3**,** nr4**,** nr5**,** nr6**:** std\_logic**;**

**signal** test1**,** test2**,** test3**,** test4**,** test5**,** test6**:** std\_logic\_vector**(**7 **downto** 0**);**

**signal** b5**,** b10**,** b20**,** b50**,** b100**,** b500**:** std\_logic\_vector**(**7 **downto** 0**);**

**signal** PIN**:** std\_logic\_vector**(**15 **downto** 0**);**

**signal** unit**,** zeci**,** sute**,** mii**,** unit1**,** zeci1**,** sute1**,** mii1**,** unit2**,** zeci2**,** sute2**,** mii2**:** integer**;**

**signal** operatie**:** std\_logic**;**

**signal** pi**,** pn**,** s1**:** std\_logic**;**

**signal** suma\_ceruta**,** pin\_nou**:** std\_logic\_vector**(**15 **downto** 0**);**

**signal** cclk**,** cclk1**,** cclk2**,** cclk3**,** cclk4**,** cclk5**:** std\_logic**;**

**begin**

D1**:** DEBOUNCER **port** **map(**clk1**,** clk**,** '0'**,** cclk1**);**

D2**:** DEBOUNCER **port** **map(**clk2**,** clk**,** '0'**,** cclk2**);**

D3**:** DEBOUNCER **port** **map(**clk3**,** clk**,** '0'**,** cclk3**);**

D6**:** DEBOUNCER **port** **map(**clk4**,** clk**,** '0'**,** cclk4**);**

D7**:** DEBOUNCER **port** **map(**clk5**,** clk**,** '0'**,** cclk5**);**

D8**:** DIVIZOR **port** **map(**clk**,** cclk**);**

C1**:** ROM\_EXP **port** **map(**'1'**,**adresa\_card**,**an\_exp**,**luna\_exp**);**

C2**:** COMP\_DATA **port** **map(** an\_exp**,** luna\_exp**,** data\_ok**);**

C3**:** RAM\_PIN **port** **map(**data\_ok**,** '0'**,** cclk**,** adresa\_card**,** pin\_nou**,** pin\_dat**);**

Z0**:** SELECTARE\_PIN\_SUME **port** **map(**cclk5**,**'1' **,**'1'**,** pi**,** pn**,** s1**);**

Z1**:** PIN\_UNIT **port** **map(**cclk1**,** pi**,** unit**);**

Z2**:** PIN\_ZECI **port** **map(**cclk2**,** pi**,** zeci**);**

Z3**:** PIN\_SUTE **port** **map(**cclk3**,** pi**,** sute**);**

Z4**:** PIN\_MII **port** **map(**cclk4**,** pi**,** mii**);**

Z5**:** ASAMBLARE\_PIN **port** **map(**unit**,** zeci**,** sute**,** mii**,** pi**,** PIN**);**

Z6**:** PIN\_UNIT **port** **map(**cclk1**,** pn**,** unit1**);**

Z7**:** PIN\_UNIT **port** **map(**cclk2**,** pn**,** zeci1**);**

Z8**:** PIN\_UNIT **port** **map(**cclk3**,** pn**,** sute1**);**

Z9**:** PIN\_UNIT **port** **map(**cclk4**,** pn**,** mii1**);**

Z10**:** ASAMBLARE\_PIN **port** **map(**unit1**,** zeci1**,** sute1**,** mii1**,** pn**,** pin\_nou**);**

Z11**:** PIN\_UNIT **port** **map(**cclk1**,** s1**,** unit2**);** --

Z12**:** PIN\_UNIT **port** **map(**cclk2**,** s1**,** zeci2**);**

Z13**:** PIN\_UNIT **port** **map(**cclk3**,** s1**,** sute2**);**

Z14**:** PIN\_UNIT **port** **map(**cclk4**,** s1**,** mii2**);**

Z15**:** ASAMBLARE\_PIN **port** **map(**unit2**,** zeci2**,** sute2**,** mii2**,** s1**,** suma\_ceruta**);**

C4**:** COMP\_PIN **port** **map(** pin\_dat**,** PIN**,** pin\_ok**);**

C5**:** DMUX\_OP **port** **map(**pin\_ok**,** '1'**,** OP**,** o1**,** o2**,** o3**,** o4**);**

C6**:** RAM\_SOLD **port** **map(**o1**,** '0'**,** cclk**,** adresa\_card**,** suma\_noua**,** suma\_existenta**);**

P1**:** DECODIFICATOR\_MII **port** **map(**suma\_existenta**,** dec\_mii**);**

P2**:** DECODIFICATOR\_SUTE **port** **map(**suma\_existenta**,** dec\_sute**);**

P3**:** DECODIFICATOR\_ZECI **port** **map(**suma\_existenta**,** dec\_zeci**);**

P4**:** DECODIFICATOR\_UNIT **port** **map(**suma\_existenta**,** dec\_unit**);**

C7**:** AFISARE\_SOLD **port** **map(**anod**,** catod**,** dec\_zeci**,** dec\_sute**,** dec\_mii**,** clk**,** o1**);**

C8**:** RAM\_PIN **port** **map(**o2**,** '1'**,** cclk**,** adresa\_card**,** pin\_nou**,** pin\_nou1**);**

C9**:** AFISARE\_PIN\_SCHIMBAT **port** **map(**anod**,** catod**,** clk**,** o2**,** eliberare\_chitanta**);**

C10**:** COMP\_1000 **port** **map(**o3**,** suma\_ceruta**,** suma\_ok**);**

C11**:** RAM\_SOLD **port** **map(**suma\_ok**,** '0'**,** cclk**,** adresa\_card**,** suma\_noua**,** suma\_exist**);**

C12**:** COMP\_SUMA **port** **map(**suma\_exist**,** suma\_ceruta**,** suma\_da**);**

C13**:** NUM\_5 **port** **map(**cclk**,** '0'**,** '0'**,** '0'**,** d5**,** n5**);**

C14**:** NUM\_10 **port** **map(**cclk**,** '0'**,** '0'**,** '0'**,** d5**,** n10**);**

C15**:** NUM\_20 **port** **map(**cclk**,** '0'**,** '0'**,** '0'**,** d5**,** n20**);**

C16**:** NUM\_50 **port** **map(**cclk**,** '0'**,** '0'**,** '0'**,** d5**,** n50**);**

C17**:** NUM\_100 **port** **map(**cclk**,** '0'**,** '0'**,** '0'**,** d5**,** n100**);**

C18**:** NUM\_500 **port** **map(**cclk**,** '0'**,** '0'**,** '0'**,** d5**,** n500**);**

C19**:** COMP\_BANC **port** **map(**suma\_da**,** suma\_ceruta**,** n5**,** n10**,** n20**,** n50**,** n100**,** n500**,** d5**,** d10**,** d20**,** d50**,** d100**,** d500**,** banc\_ok**);**

C20**:** NUM\_5 **port** **map(**cclk**,** banc\_ok**,** '0'**,** '1'**,** d5**,** test1**);**

C21**:** NUM\_10 **port** **map(**cclk**,** banc\_ok**,** '0'**,** '1'**,** d10**,** test2**);**

C22**:** NUM\_20 **port** **map(**cclk**,** banc\_ok**,** '0'**,** '1'**,** d20**,** test3**);**

C23**:** NUM\_50 **port** **map(**cclk**,** banc\_ok**,** '0'**,** '1'**,** d50**,** test4**);**

C24**:** NUM\_100 **port** **map(**cclk**,** banc\_ok**,** '0'**,** '1'**,** d100**,** test5**);**

C25**:** NUM\_500 **port** **map(**cclk**,** banc\_ok**,** '0'**,** '1'**,** d500**,** test6**);**

C26**:** SCAZATOR\_SOLD **port** **map(**banc\_ok**,** suma\_ceruta**,** suma\_exist**,** suma\_noua**,** scad\_ok**,** eliberare\_numerar**,** eliberare\_card**);**

C27**:** RAM\_SOLD **port** **map(**scad\_ok**,** '1'**,** cclk**,** adresa\_card**,** suma\_noua**,** suma\_exist1**);**

C28**:** DECOD\_BANC **port** **map(**o4**,** bancnota\_introdusa**,** banc\_decod**,** bancnota\_ok**);**

C29**:** RAM\_SOLD **port** **map(**o4**,** '0'**,** cclk**,** adresa\_card**,** suma\_noua**,** suma\_exist2**);**

C30**:** SUMATOR\_SOLD **port** **map(**banc\_decod**,** suma\_exist2**,** sold\_nou**);**

C31**:** RAM\_SOLD **port** **map(**o4**,** '1'**,** cclk**,** adresa\_card**,** sold\_nou**,** sold\_exist1**);**

C32**:** DMUX\_BANC **port** **map(**'1'**,** o4**,** banc\_decod**,** nr1**,** nr2**,** nr3**,** nr4**,** nr5**,** nr6**);**

C33**:** NUM\_5 **port** **map(**cclk**,** nr1**,** '0'**,** '0'**,** d5**,** b5**);**

C34**:** NUM\_10 **port** **map(**cclk**,** nr2**,** '0'**,** '0'**,** d5**,** b10**);**

C35**:** NUM\_20 **port** **map(**cclk**,** nr3**,** '0'**,** '0'**,** d5**,** b20**);**

C36**:** NUM\_50 **port** **map(**cclk**,** nr4**,** '0'**,** '0'**,** d5**,** b50**);**

C37**:** NUM\_100 **port** **map(**cclk**,** nr5**,** '0'**,** '0'**,** d5**,** b100**);**

C38**:** NUM\_500 **port** **map(**cclk**,** nr6**,** '0'**,** '0'**,** d5**,** b500**);**

**end** STRUCTURALA**;**

Prima parte (C1 -> C23) consta in indentificarea cardului si verificarea pinului: se extrage din memoria ROM data expirarii si se compara cu data introdusa de utilizator, apoi se extrage din memoria RAM pinul si se compara cu cel introdus de utilizator. Daca aceste operatii se efectueaza cu succes urmeaza alegerea operatiei dorite cu ajutorul demultiplexorului.

Partea de interogare sold(C24 -> C29) este alcatuita din extragerea din memoria RAM a soldului, decodificarea si afisarea acestuia.

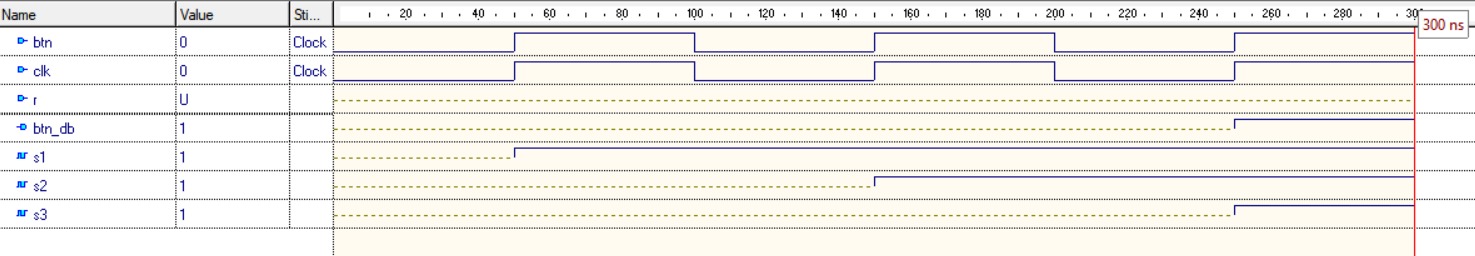
Partea de schimbare pin(C30 -> C31) este alcatuita din extragerea din memoria RAM a pinului si afisarea acestuia.

Partea de extragere numerar(C32 -> C49) este alcatuita din: comparatorul cu 1000 de euro care compara suma ceruta cu suma maxima admisa pentru extragere, extragerea soldului din memoria RAM si compararea acestuia cu suma ceruta de utilizator, cele 5 numaratoare pentru fiecare tip de bancnota, verificarea printr-un comparator a tipurilor de bancnote existente si daca toate aceste operatii sunt efectuate cu succes urmeaza scaderea soldului, adaugarea acestuia in memoria RAM si scaderea numarului de bancnote.

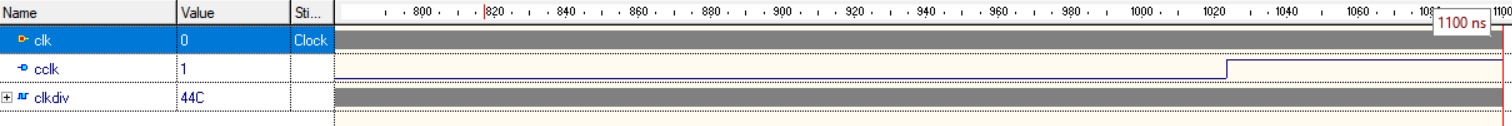
Partea de depunere numerar(C50 -> C60) este alcatuita din: intai un decodificator pentru bancnote, un sumator pentru adunarea sumei introduse si apoi adaugarea acesteia in memoria RAM, apoi un demultiplexor pentru selectarea tipului de bancnote introduse si cele 5 numaratoare pentru a se putea realiza depunerea, numararea banilor cu succes.

**Simularea componentelor si a codului ansamblu:**

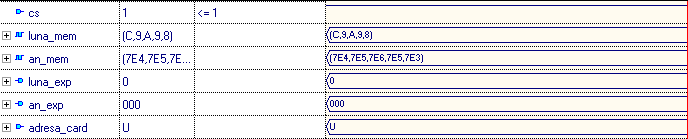
1. Debouncer (realizat de Corpodean Darius)



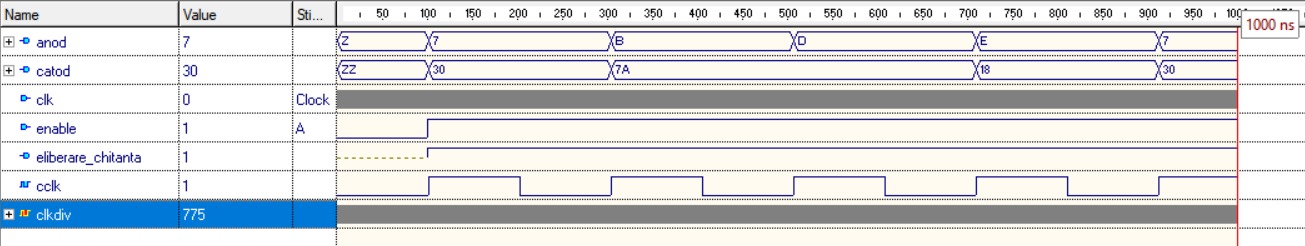
1. Divizor frecventa (realizat de Corpodean Darius)



1. Memorie ROM Data Expirare card (realizat de Corpodean Darius)

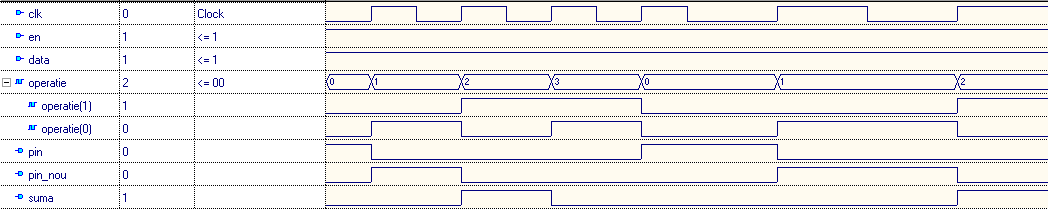


1. Afisare eroare card (realizat de Coblisan George)



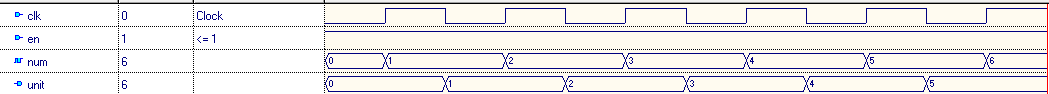
Am simulat afisarea comentand divizorul de frecventa pentru a nu utiliza prea multi pasi si sa demonstrez functionalitatea sa.

1. Selectare PIN (realizat de Corpodean Darius)

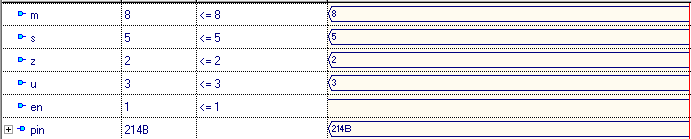


1. Numarator PIN (realizat de Coblisan George)

Am simulat un singur numarator deoarece toate au aceeasi functionalitate.

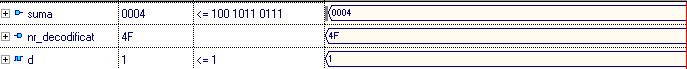


1. Formare PIN (realizat de Coblisan George)



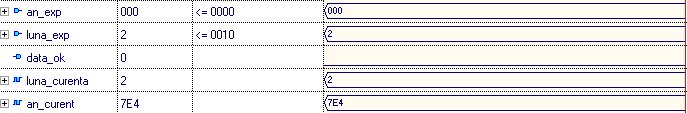
1. Decodificator mii (realizat de Corpodean Darius)



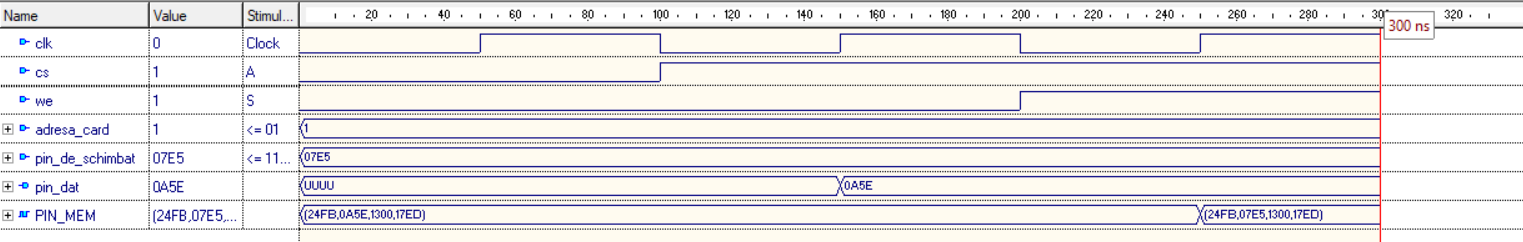


La fel, am simulat un singur decodificator deoarece toate au aceeasi functionalitate.

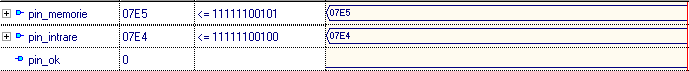
1. Comparator data curenta cu data expirarii (realizat de Corpodean Darius)



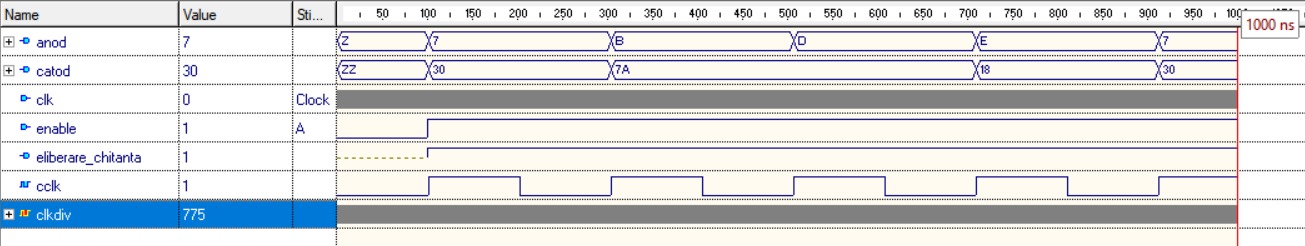
1. Memorie RAM PIN (realizat de Corpodean Darius)

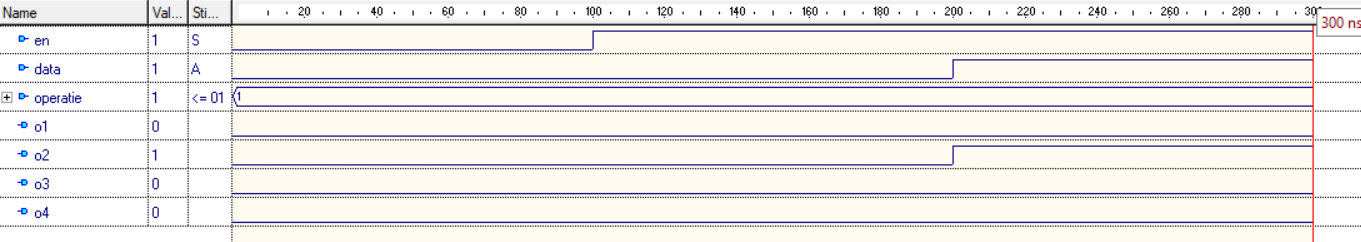


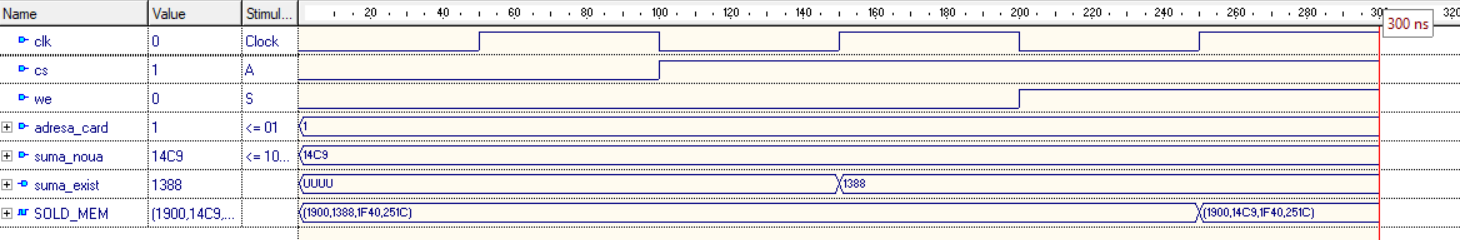
1. Comparator PIN (realizat de Corpodean Darius)



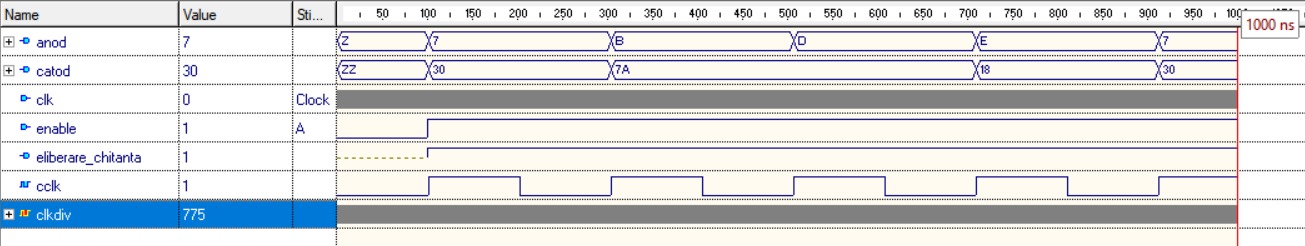
1. Afisare Eroare PIN (realizat de Coblisan George)



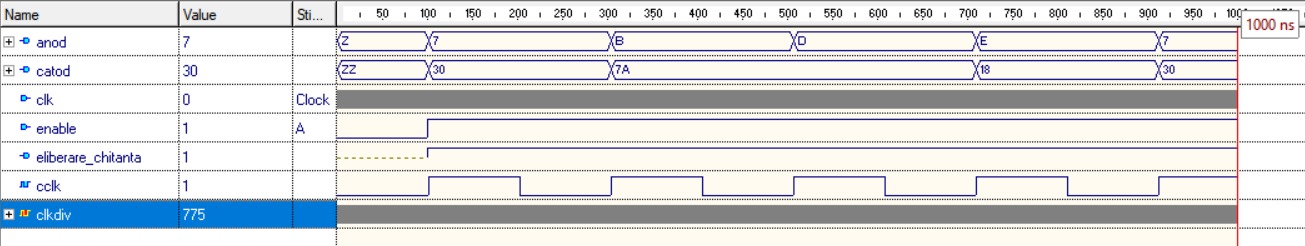
1. Demultiplexor Operatii (realizat de Corpodean Darius)
2. Memorie RAM SOLD (realizat de Coblisan George)

****

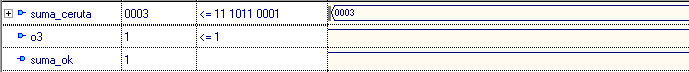
1. Afisare PIN schimbat (realizat de Coblisan George)



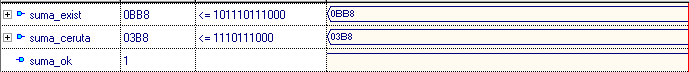
1. Afisare SOLD (realizat de Coblisan George)



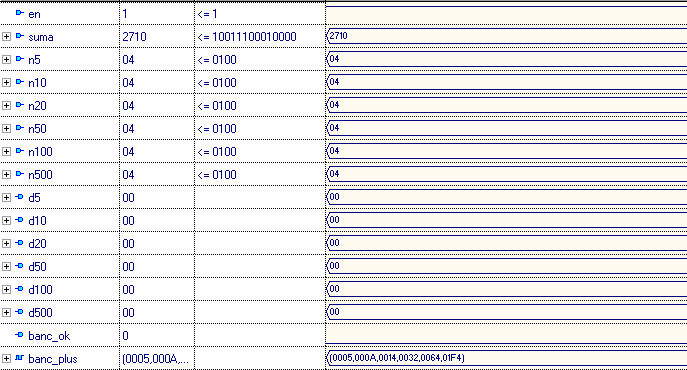
1. Comparator 1000 EURO (realizat de Coblisan George)



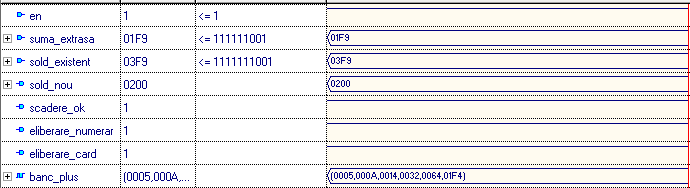
1. Comparator cu SUME (realizat de Coblisan George)



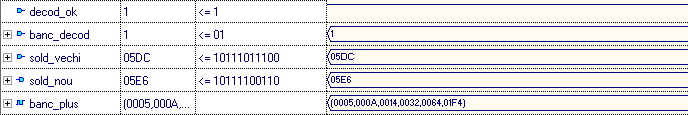
1. Comparator existenta bancnote (realizat de Corpodean Darius)



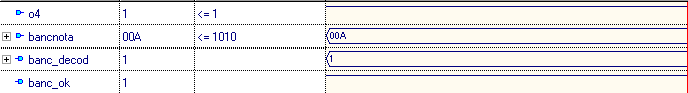
1. Scazator Memorie RAM SOLD (realizat de Coblisan George)



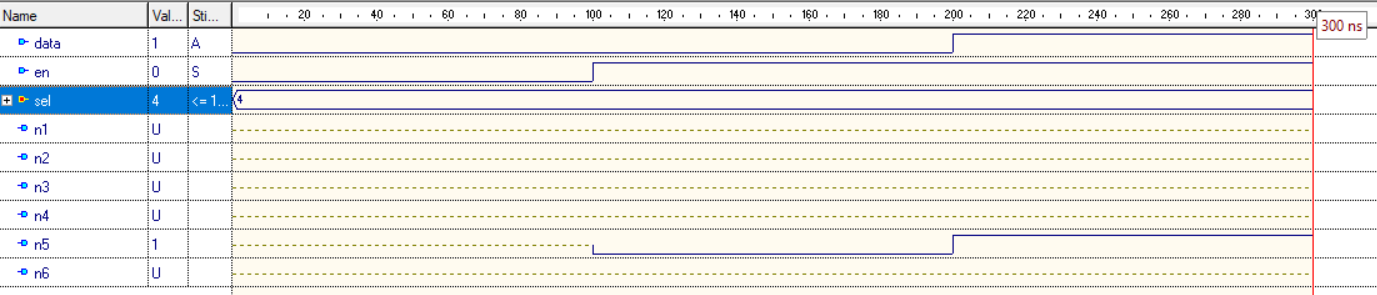
1. Sumator Memorie RAM (realizat de Corpodean Darius)



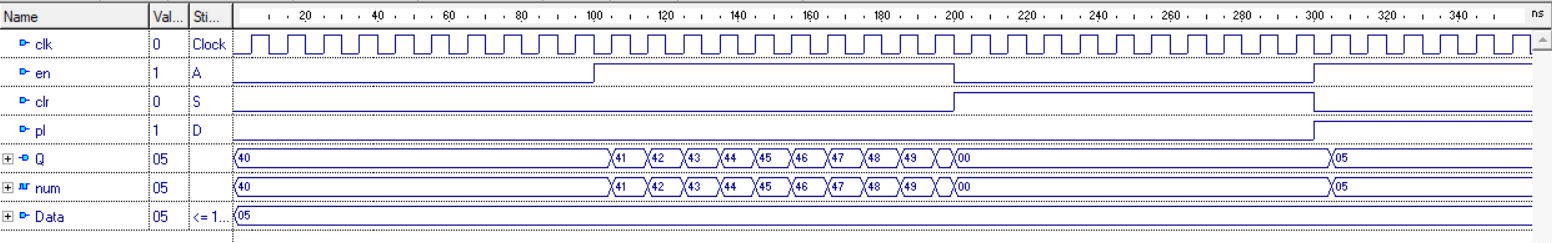
1. Decodificator Bancnote (realizat de Corpodean Darius)



1. Demultiplexor Bancnote (realizat de Corpodean Darius)

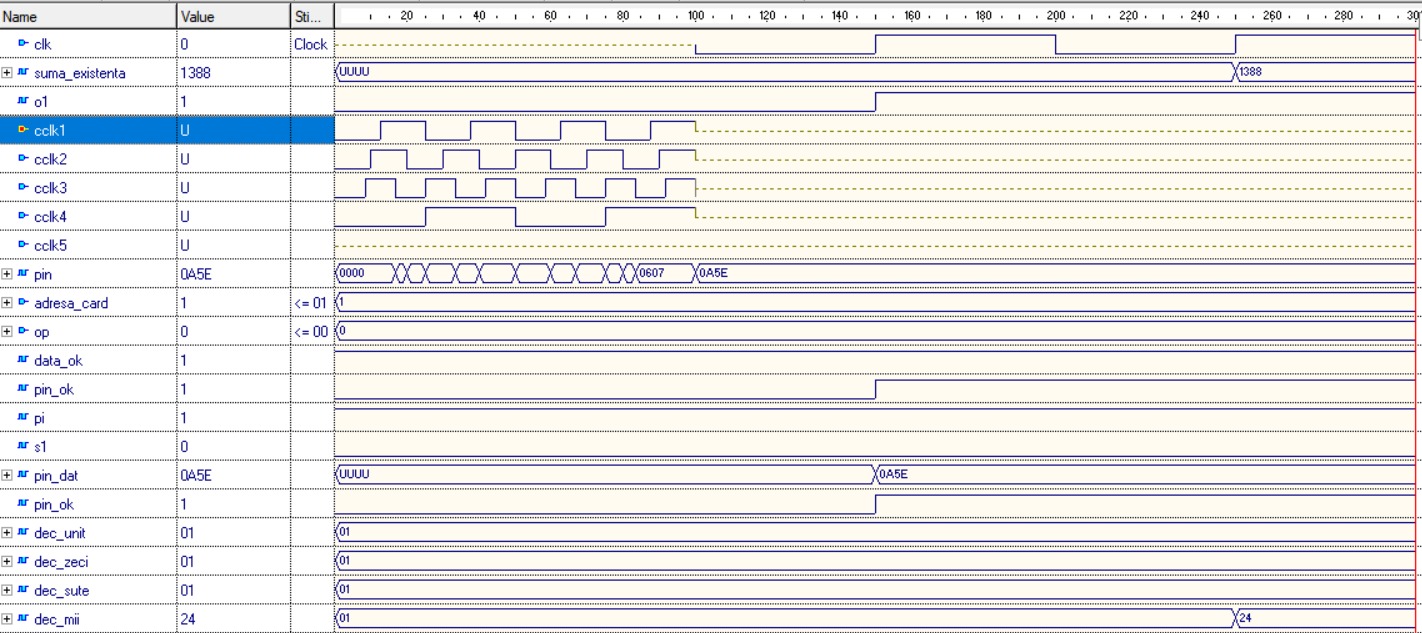


1. Numarator 5 euro (realizat de Corpodean Darius)

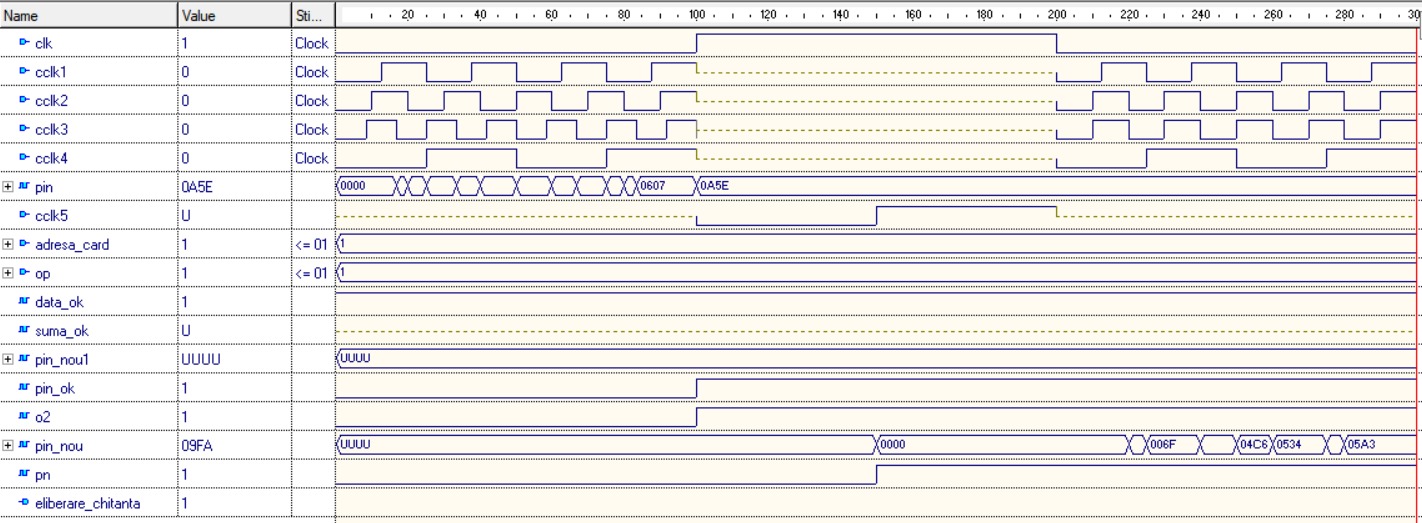
****

Am simulat un singur numerator pentru ca toate functioneaza exact la fel doar ca sunt folosite pentru valori de bancnote diferite.

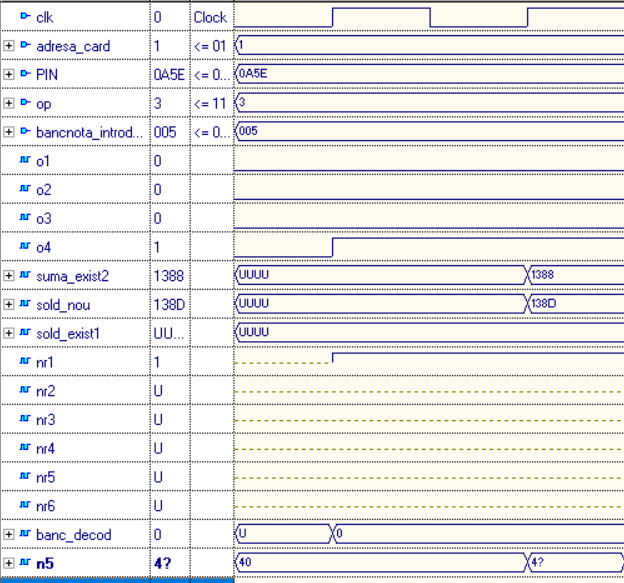
Operatia 1 – Interogare sold (realizata de Coblisan George)

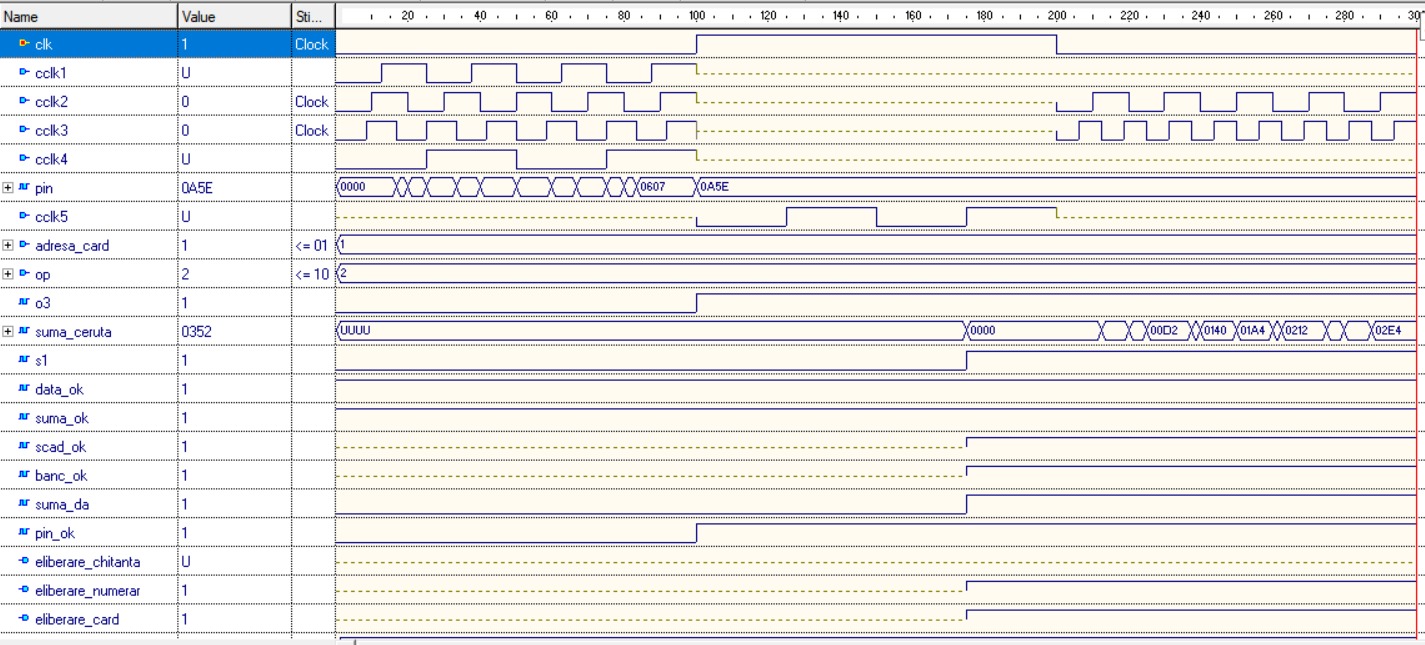


Operatia 2 – schimbare pin (realizata de Coblisan George)

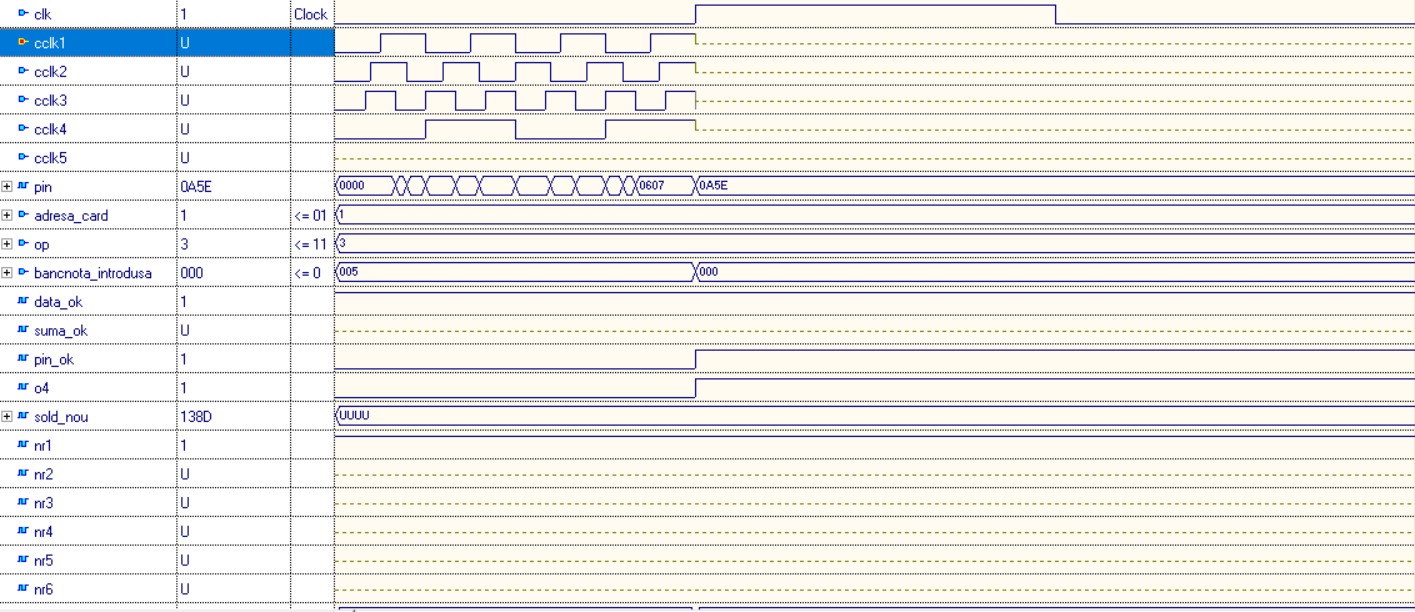


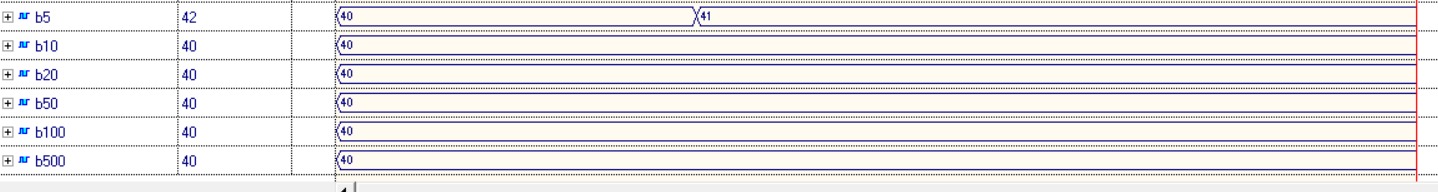
Operatia 3 – extragere numerar (realizata de Corpodean Darius)



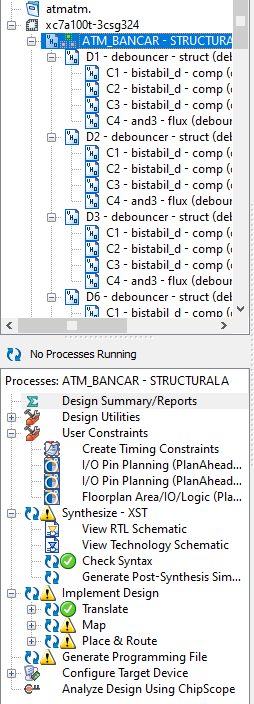


Operatia 4 – depunere numerar (realizata de Corpodean Darius)





**Fisierul de constrangeri in ISE (realizat de Coblisan George)**





1. **Lista de componente utilizate (capitol realizat de Coblisan George)**

* Memorie ROM pentru data expirarii cardului.
* Memorie RAM pentru sold si pentru pin.
* Comparatoare pentru data expirarii, suma ceruta, 1000 de EURO, PIN.
* Demultiplexoare 1:4 si 1:8 pentru selectarea operatiei, selectarea bancnotelor.
* Decodificatoare pentru bancnote si pentru sold.
* Scazatoare pentru operatia de extragere (sold si bancnote).
* Sumator pentru depunere.
* Numaratoare pentru bancnote si pentru introducerea pinului.

1. **Semnificatia notatilor de I/O si a semnalelor interne (capitol realizat de Coblisan George)**

Intrari:

* CLK – clockul mare de pe placa.
* CLK1 – clock (de la buton) pentru cresterea cifrei unitatilor a pinului.
* CLK2 – clock (de la buton) pentru cresterea cifrei zecilor a pinului.
* CLK3 – clock (de la buton) pentru cresterea cifrei sutelor a pinului.
* CLK4 – clock (de la buton) pentru cresterea cifrei miilor a pinului.
* CLK5 – clock (de la buton) pentru componenta de selectare a pinului, pinului nou sau sumei.
* Adresa\_card – vector de 3 biti in care se retine adresa cardului utilizat.
* OP – vector de 2 biti care reprezinta operatia aleasa.
* Bancnota\_introdusa – vector de 9 biti pentru a identifica tipul bancnotei introduse, folosita doar la operatia de depunere.

Iesiri:

* Anod, Catod – pentru afisare.
* Eliberare\_card, eliberare\_chitanta, eliberare\_numerar.

Semnale interne:

* An\_exp – vector de 11 biti pentru anul expirarii cardului.
* Luna\_exp – vector de 4 biti pentru luna expirarii cardului.
* Data\_ok – ne spune daca cardul este expirat sau nu.
* Suma\_ok – ne spune daca suma introdusa poate fi extrasa si daca exista pe card.
* Suma\_da – daca exista pe card, enable pentru scaderea bancnotelor.
* Banc\_ok – ne spune daca exista bancnotele necesare pentru extragerea sumei.
* Scad\_ok – ne spune daca scaderea soldului a fost efectuata cu succes.
* Bancnota\_ok – ne spune daca decodificarea bancnotelor a fost efectuata cu succes.
* Pin\_dat – pinul iesit din memoria RAM si folosit la compararea celui vechi cu cel nou.
* Pin\_nou – pinul care trebuie schimbat in memoria RAM si pinul asamblat.
* Pin\_nou1 – pinul schimbat dupa operatia 2.
* Pin\_ok – ne spune daca pinul a fost schimbat cu succes si activeaza demultiplexorul pentru selectarea operatiei.
* PIN – pinul rezultat dupa asamblare.
* O1, O2, O3, O4 – cele 4 operatii: interogare sold, schimbare pin, extragere numerar, depunere numerar.
* Suma\_exist – iesirea memoriei RAM pentru sold, in care e retinut soldul total de pe card.
* Suma\_noua – introduce noua suma dupa extragere in memorie.
* Sold\_nou – retinut soldul nou dupa operatia de adunare.
* Suma\_existenta – suma folosita pentru decodificare.
* Suma\_ceruta – suma ceruta de utilizator, folosita pentru comparare.
* Suma\_exist1 – soldul existent dupa operatia de extragere.
* Suma\_exist2- suma de bani folosita la operatia de depunere.
* Sold\_exist1- soldul existent dupa operatia de depunere.
* Dec\_unit, dec\_zeci, dec\_sute, dec\_mii – suma de bani decodificata.
* N5, N10, N20, N50, N100, N500 – sumele iesite din numaratoare.
* D5, D10, D20, D50, D100, D500 – datele introduse in numaratoare (pentru incarcare paralela).
* Banc\_decod – bancnota decodificata.
* NR1, NR2, NR3, NR4, NR5, NR6 – iesirile demultiplexorului pentru bancnote si enable pentru numaratoarele pentru bancnote.
* Test1, test2, test3, test4, test5, test6 – iesirile numaratoarelor dupa partea de extragere numerar.
* B5, B10, B20, B50, B100, B500 – iesirile numaratoarelor dupa partea de depunere numerar.
* Unit, zeci, sute, mii – unitatile, zecile, sutele si miile folosite pentru asamblarea pinului / sumei cerute / pinului nou.
* Pi, pn, s1, - pinul, pinul nou si suma ceruta rezultate dupa operatia de selectare.
* Decod\_ok – enable pentru sumatorul soldului.

1. **Justificarea solutiei alese (capitol realizat de amandoi)**

Am optat pentru aceasta solutie deoarece ni s-a parut cea mai apropiata de realitate si cea mai usor de inteles deoarece ordinea in care se trece de la o componenta la alta este cat se poate de logica.

Ne-am dorit sa implementam un sistem cat mai usor de folosit pentru utilizatori si sa selectam strict cele mai uzuale operatii intalnite la un bancomat din viata de zi cu zi, dar si cele mai utile, astfel incat proiectul nostru sa respecte in proportie mare ideea pe care toti o avem cand ne gandim la un automat bancar.

Folosim bancomatele in viata noastra de zi cu zi, si prin urmare este normal sa incercam sa intelegem functionarea acestora si la randul nostru sa le putem implementa astfel incat sa facilitam utilizatorii care le vor folosi la randul lor.

O problema initiala ne-a fost creata de numarul mare de porturi de intrare, un numar de porturi apropiat de 60 de intrari diferite. Un bancomat astazi utilizeaza un numar maxim de 20-22 de butoane, deci maxim 22 de intrari diferite si asta s-a dovedit o problema de princpiu in solutia aleasa de noi. Prin urmare, am abordat o solutie care ni s-a parut foarte aproape de realitate si totodata usor de utilizat astfel:

Vechile intrari pentru introducerea pinului si eventual in caz de necesitate a sumei necesare pentru extragere sau a unui pin nou se foloseau fiecare de 16 intrari diferite, insumate ajungand la 48 de intrari separate (fiecare reprezenta un vector de 16 biti si incurca, nu doar atat echipa de implementare cat si pe utilizator avand in vedere ca numerele in baza 2 mari sunt extrem de greu de retinut).

Solutia acestei probleme a fost reducerea de la 48 de intrari separate la doar 5 prin implementarea unor butoane care sa functioneze pentru mai multe intrari: primul buton selecteaza ce dorim sa introducem (neapasat se poate introduce pinul, apasat o data se introduce pinul nou si daca il apasam de 3 ori se va introduce suma ceruta).

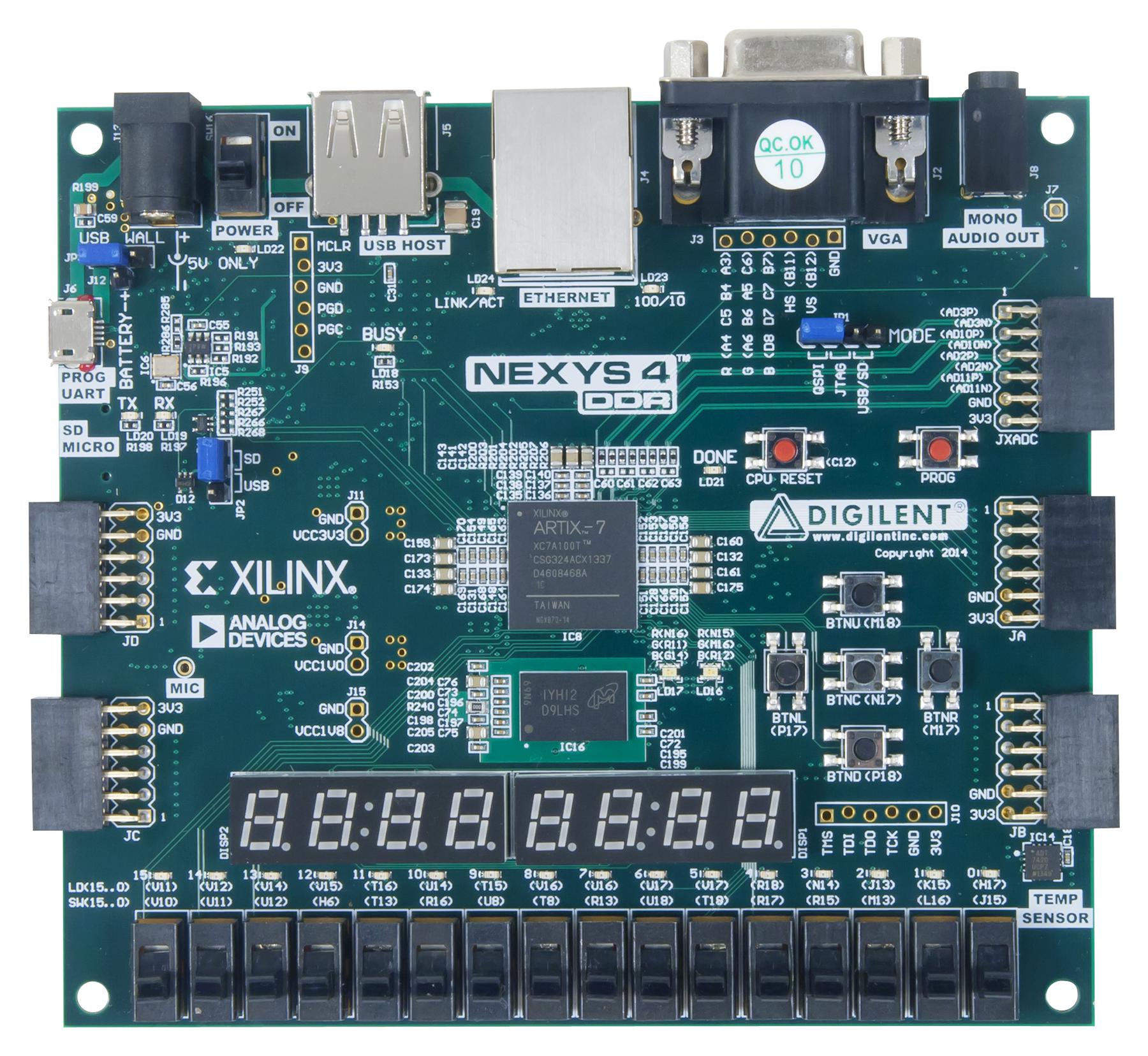
Cele 3 se introduc mult mai usor decat inainte: exista 4 butoane (cate unul pentru mii, sute, zeci, unitati) si prin urmare de cate ori e apasat fiecare buton, aceea va fi cifra acelui ordin:

Apasam de 2 ori primul buton, de 5 ori pe al doilea, de 7 ori pe al treilea si de 9 ori pe ultimul. Numarul format va fi 2579.

Aceasta metoda de implementare dupa cum ati putut vedea este una dintre cele mai bune metode si cea mai buna gasita de noi.

1. **Utilizare si rezultate (capitol realizat de Corpodean Darius)**

Resursele necesare pentru realizarea proiectului nostru au fost Active HDL, simulatorul din Active HDL, editorul schematic si ISE Design Suite.



Placa aleasa de noi care corespunde necesatilor proiectului este Nexys 4.

Aceasta dispune de 8 afisoare 7 segmente, 16 comutatoare cu 2 stari, 5 comutatoare de tip push – botton si LED-urile.

Dintre cele 16 comutatoare cu 2 stari:

* 3 au fost folosite pentru introducerea adresei cardului
* 2 pentru selectarea operatiei
* 9 pentru introducerea bancnotei

Dintre cele 5 comutatoare de tip push-botton:

* Unul este folosit pentru selectarea pinului / pinului nou / sumei cerute
* Celelalte 4 sunt folosite pentru determinarea unitatilor, zecilor, sutelor si miilor numarului selectat mai sus

Clockul integrat pe placa FPGA

Placa FPGA Nexys 4 include un oscliator de quartz de 100 MHz care se gaseste la locatia E3 pe placa. Acest clock este conectat la un divizor de frecventa si prin urmare permite sistemului sa functioneze la intervale de timp care permit utilizatorului sa vizualizeze buna functionare a sistemului.

1. **Posibilitati de dezvoltare ulterioara (capitol realizat de Corpodean Darius)**

In starea in care se afla proiectul in momentul actual, acesta se poate dezvolta fara foarte multe fonduri investite sau multa munca in plus. Baza (back- endul) si functioarea optima a acestuia exista. Ulterior ar putea fi implementate senzori pentru recunoasterea cardului(cititoare de chip-uri, cititoare NFC), pentru a nu trebui introdusa adresa cardului si cautata intr-o baza de date relativ mare dupa un ID introdus fapt ce scuteste utilizatorul de o munca in plus.

Dezvoltarea acestui proiect ar consta in realizarea unui bancomat la fel de util ca cele de azi in cel mai rau caz. In cel mai bun caz dezvoltarea acestuia ar trebui sa surclaseze bancomatele de azi si sa ii confere utilizatorului atat o utilizare imbunatatita, cat si una mai sigura si mai placuta.

Aceasta ar putea fi realizata prin introducerea cititoarelor NFC si astfel o sa il scuteasca pe client de introducerea cardului si asteparea pentru eliberarea sa, si totodata un pin introdus gresit, care poate sa se intample oricaruia dintre noi, blocheaza cardul in automatul bancar fapt extrem de neplacut in realitatea in care traim timpul fiind un lux pe care unii din oameni nu il au din cauza unui stil de viata aglomerat. Cititorul poate sa sesiseze pinul introdus gresit si sa notifice posesorul de card instant, datorita noilor tehnologii de pe piata si a vitezei de internet superior, rezolvand astfel problema securitatii cardului. Modulul ce contine proiectul nostru ar trebui in acest caz echipat cu un citior NFC si un modul wi-fi sau de date mobile care sa poata sa il tina informat pe client.

Totodata, proiectul va fi dezvoltat ulterior prin aducerea echipamentului necesar precum: un display pentru afisarea informatiei(care ar putea fi capacitiv) cu o interfata usor de navigat, in locul afisoarelor 7-segmente si ale ledurilor, a butoanelor programate special pentru a nu mai fi nevoiti sa apasam un buton de 5 ori pentru cifra 5 ci doar odata butonul 5 pentru a o obtine.

Am putea dota automatele si cu un difuzor pentru a ghida prin mesaje o persoana cu dizabilitati de vedere, care ar fi un cost scazut si ar ajuta o gama destul de larga de persoane.

Aceste dezvoltari se pot face pe o placuta FPGA cat si pe un alt tip de placuta, cele FPGA fiind costisitoare, foarte bune pentru implementare unui proiect, nu atat de bune pentru dezvoltarea la scara. In cazul in care am continua acest proiect lucurile specificate ar fi primele planuri pentru dezvoltarea ulterioara a proiectului nostru.