MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

CIFRU DULAP

PROIECTAREA SISTEMELOR NUMERICE

Studenți: Stoica Mihai, Sărăndan Vlad

Coordonator: Szabolcs - Andras Csillag

2024

Cuprins

1.Specificație proiect

Descriere: Să se implementeze un sistem numeric care permite utilizatorului adăugarea unui cifru din 3 caractere distincte pentru securizarea unui dulap (asemănător dulapurilor folosite la vestiarele de la sălile de sport, mall, etc)

Cerințe funcționale:

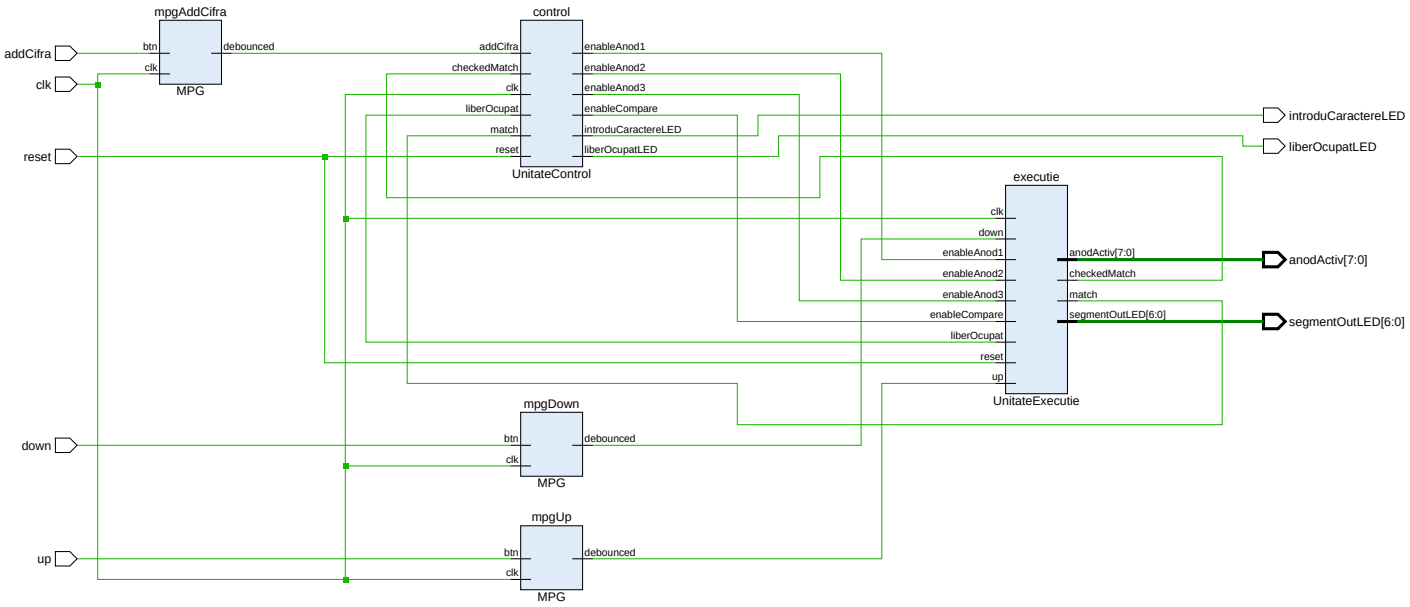
1. Un led **LIBER_OCUPAT** va avea funcția de a semnala faptul ca dulapul este liber(led stins) sau ocupat(led aprins).
2. Utilizatorul va apasă un buton **ADAUGA_CIFRA** pentru a semnala începerea introducerii codului. Un led **INTRODU_CARACTERE** se va aprinde pentru a marca starea.
3. Utilizatorul va adaugă pe rând 3 caractere cu ajutorul butoanelor **UP** si **DOWN.**
4. Caracterele permise sunt curprinse in intervalul 0-1-..-8-9-A-B-..-F
5. Caracterul curent introdus este afișat pe afișor cu 7 segmente (SSD).
6. Pentru trecerea la următorul caracter utilizatorul va apasă butonul **ADAUGA_CIFRA.**
7. Caracterul anterior introdus rămâne afișat.
8. Următorul caracter este vizibil pe afișaj pe poziția următoare.
9. După introducerea celui de al treilea caracter, la apăsarea butonului **ADAUGA_CIFRA**, afișajul SSD se va stinge iar cifru va fi in starea blocat prin aprinderea ledului **LIBER_OCUPAT**.
10. Ledul **INTRODU_CARACTERE** se va stinge
11. Existenta unui buton/switch **RESET** in timpul introducerii cifrului pentru revenire in starea inițiala(ledul **LIBER_OCUPAT** se va stinge, afișajul SSD este gol, ledul **INTRODU_CARACTERE** se va stinge)
12. Utilizatorul va apasă butonul/switch **ADAUGA_CIFRA** pentru a începe introducerea codului pentru deblocarea cifrului
13. Se vor relua pașii 2-8.
14. La introducerea ultimului caracter, la apăsarea butonului **ADAUGA_CIFRA** se va face verificarea, daca codul introdus corespunde cu codul anterior.
15. In cazul de egalitate, ledul **LIBER_OCUPAT** se va stinge, ledul **INTRODU_CARACTERE** se va stinge, afișajul SSD se golește.
16. In cazul de inegalitate, ledul **LIBER_OCUPAT** va rămâne aprins, ledul **INTRODU_CARCATERE** se va stinge, afișajul SSD se golește.

Cerințe non-funcționale:

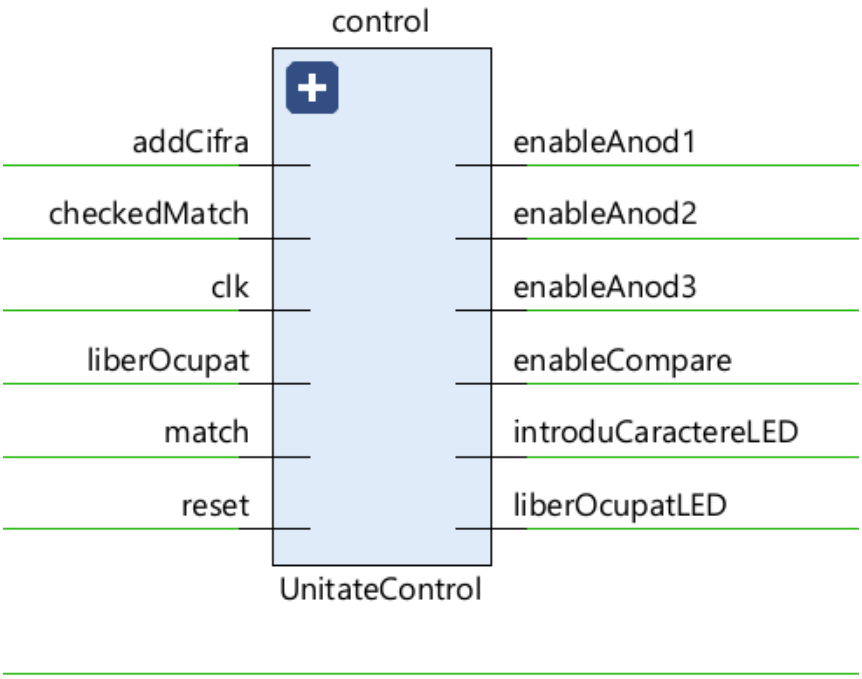
1. Implementare pe plăcuță
2. Utilizare SSD
3. Utilizare switch, led, butoane

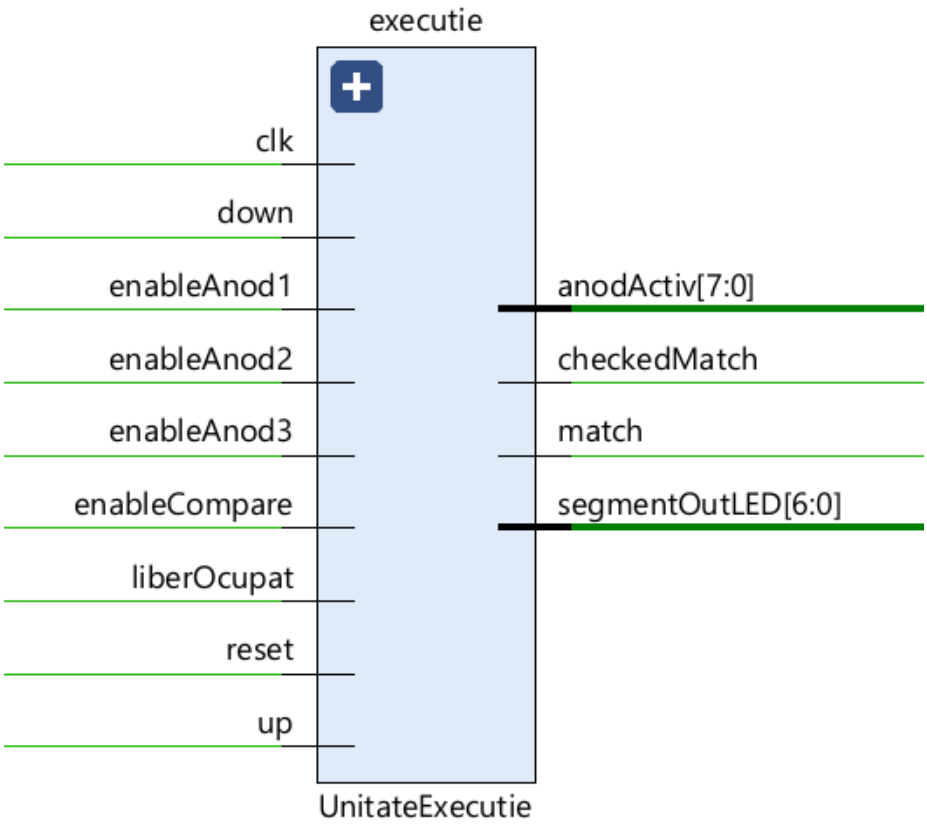
2. Schema bloc cu componentele principale

Black Box -ul conține 5 intrări: buton addCifra, up, down, clk și un switch reset. De asemenea acesta conține și 4 ieșiri: 3 anozi folosiți pentru reprezentarea pe afișorul cu 7 segmente și un led LIBER_OCUPAT care arată starea în care se află cifrul.



3.Evidenţierea unităţii de comandă şi a celei de execuţie





4.Lista componentelor folosite

```
component UnitateControl is
    port (
        clk : in std_logic;
        reset : in std_logic;
        addCifra : in std_logic;
        enableCompare : out std_logic;
        checkedMatch : in std_logic;
        enableAnod1 : out std_logic;
        enableAnod2 : out std_logic;
        enableAnod3 : out std_logic;
        match : in std_logic;

        liberOcupat : inout std_logic;
        liberOcupatLED : out std_logic;
        introduCaractereLED : out std_logic
    );
end component;
```

clk=semnal de ceas
reset=semnal de resetare pentru inițializarea componentei
addCifra=semnal care indică introducerea unei noi cifre
enableCompare=semnal de ieșire pentru inițierea unei comparații
checkedMatch=semnal care verifică efectuarea unei comparații
enableAnod1, enableAnod2, enableAnod3=semnale pentru controlul segmentelor SSD
match=indică dacă comparația a reușit
liberOcupat=ieșire care reflectă starea curentă
liberOcupatLED=ieșire pentru un LED pentru a indica vizual starea ocupată
introduCaractereLED =ieșire pentru un LED pentru a indica introducerea cifrei

```
component UnitateExecutie is
    port (
        clk : in std_logic;
        reset : in std_logic;
        liberOcupat : in std_logic;
        enableAnod1 : in std_logic;
        enableAnod2 : in std_logic;
        enableAnod3 : in std_logic;
        up : in std_logic;
        down : in std_logic;
        enableCompare : in std_logic;
        checkedMatch : out std_logic;
        match : out std_logic;

        anodActiv : out std_logic_vector (7 downto 0);
        segmentOutLED : out std_logic_vector (6 downto 0)
    );
end component;
```

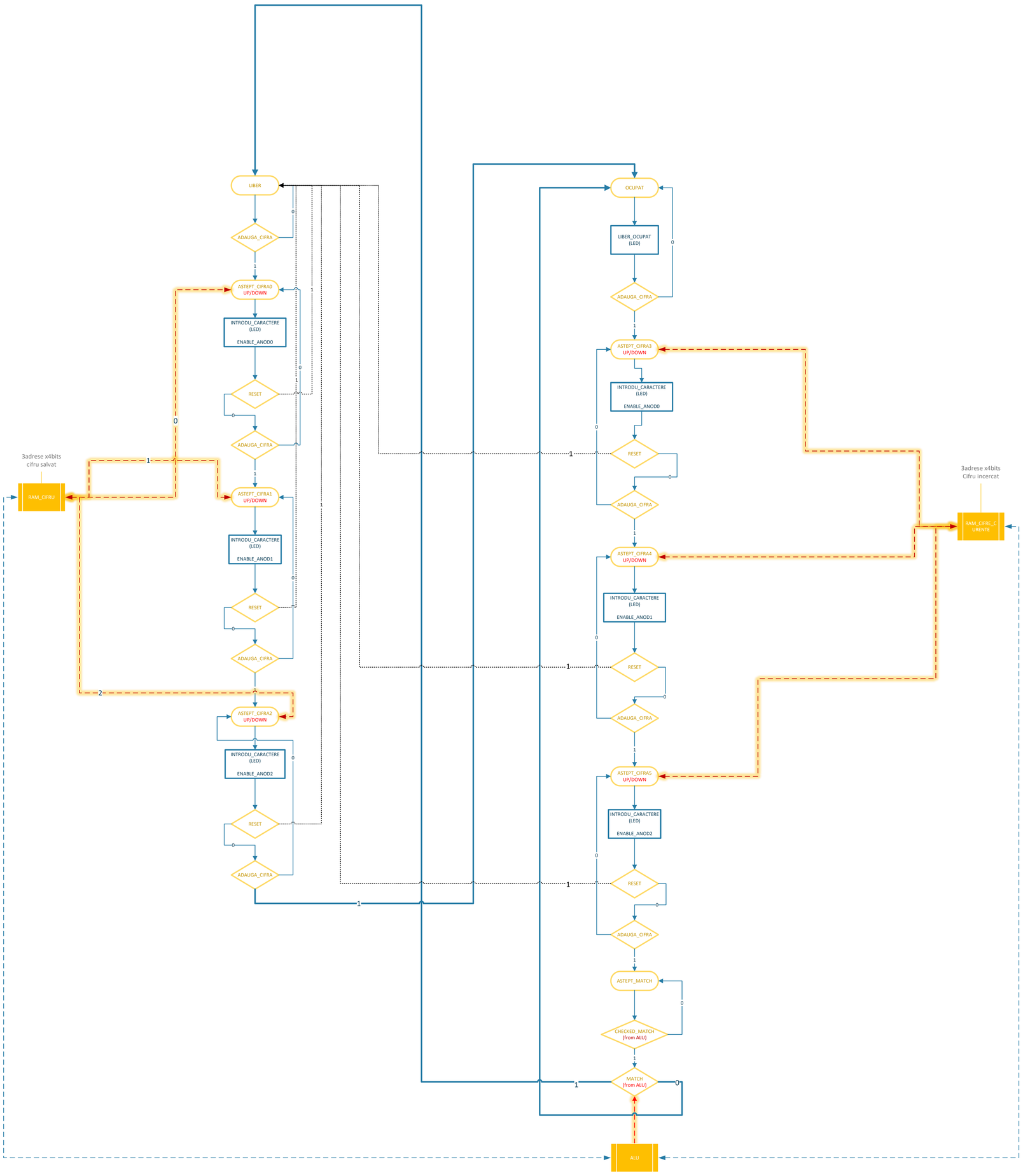
up=incrementează cifra
down=decrementează cifra

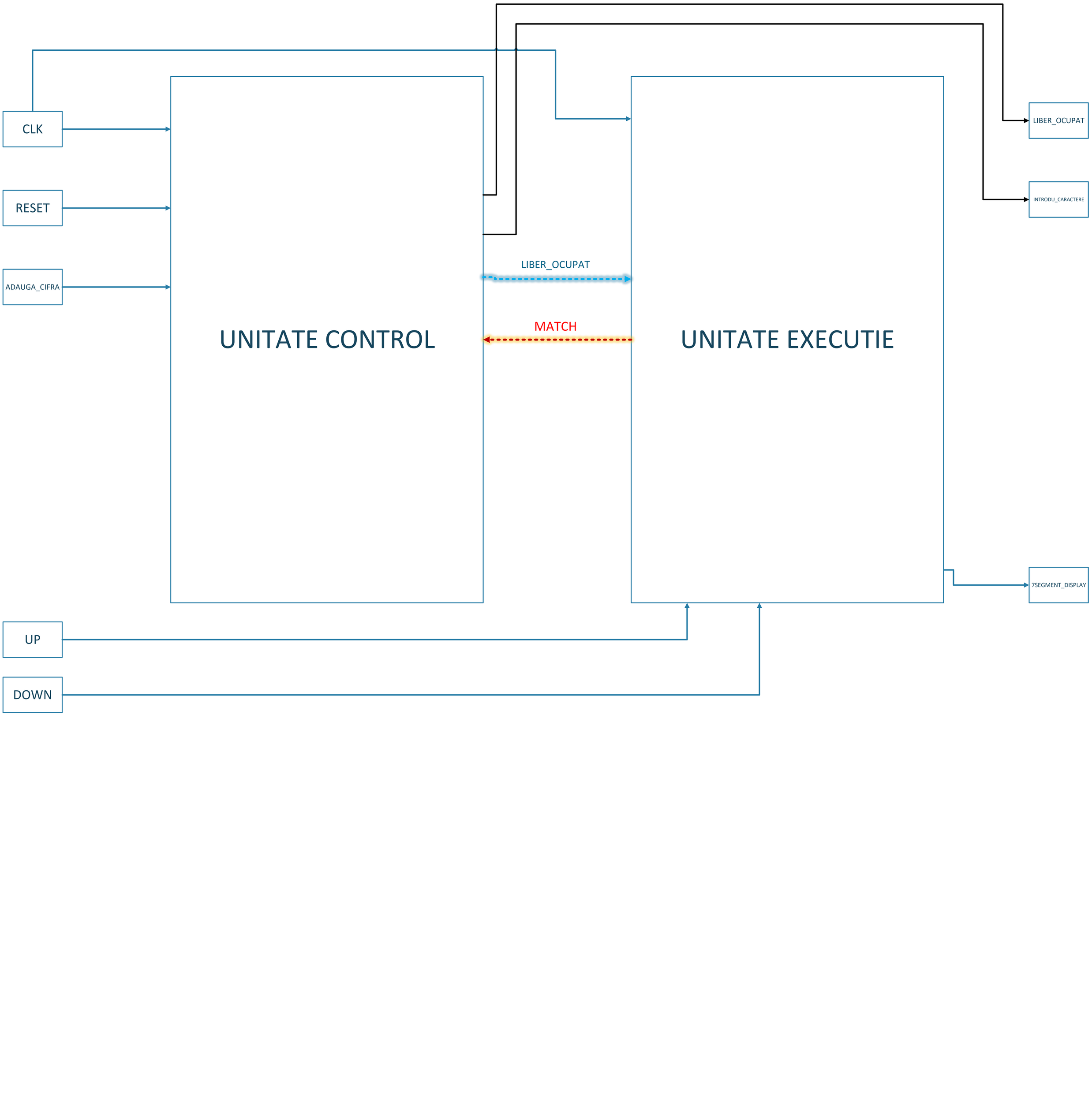
```
component MPG is
    port (
        btn : in std_logic;
        clk : in std_logic;
        debounced : out std_logic);
end component;
```

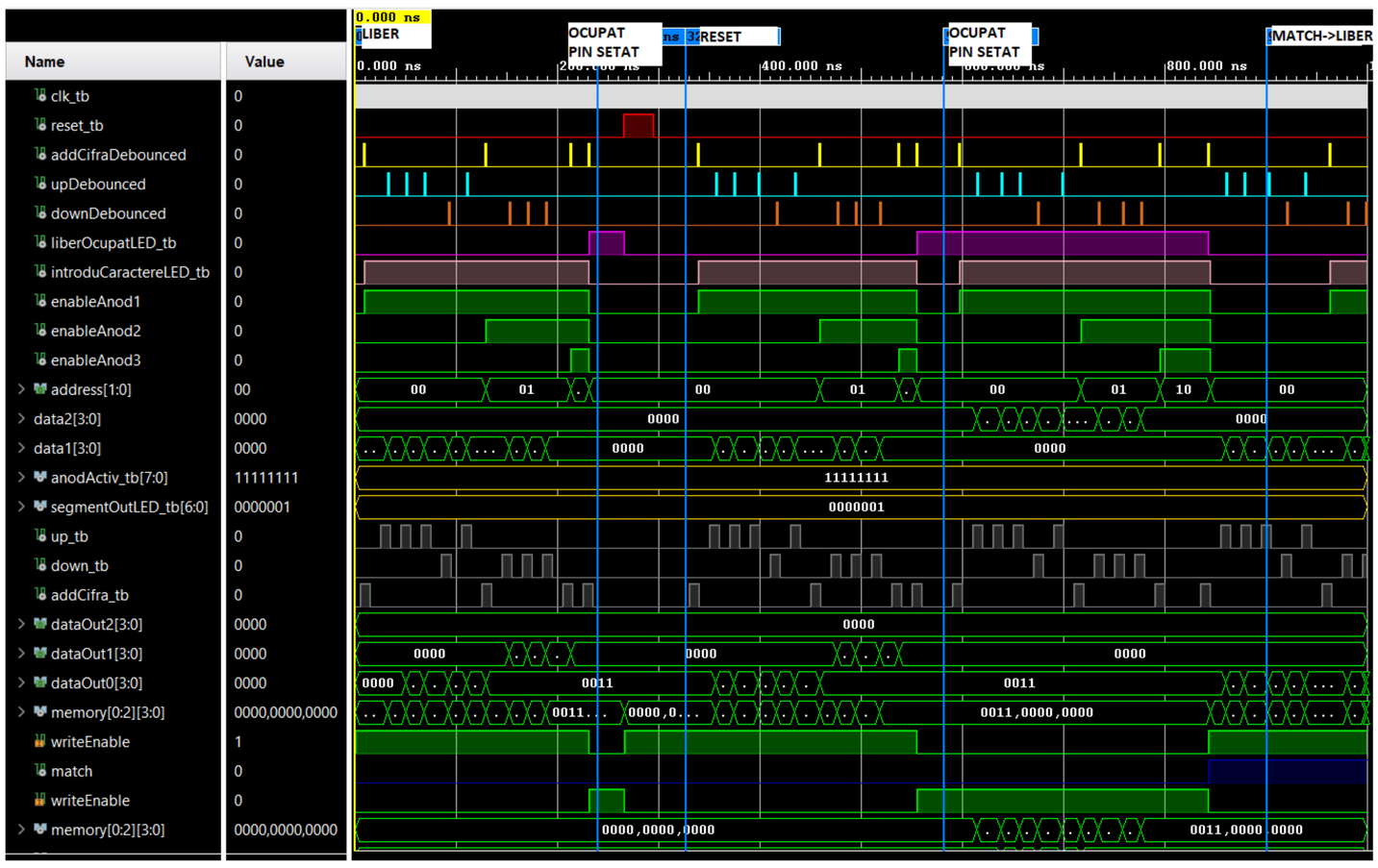
debounced=divizor de frecvență

displayController - trimite catre 7 segment cifrele curente ce trebuie afisate
ramController - selecteaza in functie de starile liber/ocupat in care dintre cele 2 memorii ram este scris pin-ul introdus
ramCifru - stocheaza pin-ul care blocheaza cifrul
ramCifreCurente - stocheaza pin-ul care se incearca pentru eliberare
comparator - compara continutul din cele doua memorii ram

ORGANIGRAMA:




CLK
RESET
ADAUGA_CIFRA
UP
DOWN
UNITATE CONTROL
LIBER_OCUPAT
MATCH
UNITATE EXECUTIE
LIBER_OCUPAT
INTRODU_CARACTERE
7SEGMENT_DISPLAY


Name
Value
clk_tb
reset_tb
addCifraDebounced
upDebounced
downDebounced
liberOcupatLED_tb
introduCaractereLED_tb
enableAnod1
enableAnod2
enableAnod3
address[1:0]
data2[3:0]
data1[3:0]
anodActiv_tb[7:0]
segmentOutLED_tb[6:0]
up_tb
down_tb
addCifra_tb
dataOut2[3:0]
dataOut1[3:0]
dataOut0[3:0]
memory[0:2][3:0]
writeEnable
match
writeEnable
memory[0:2][3:0]
0.000 ns
LIBER
OCUPAT
PIN SETAT
RESET
MATCH->LIBER
400.000 ns
800.000 ns

NEXYS 4
DDR
DIGILENT
UP
OK
DOWN
BTNU (M18)
BTNC (N17)
BTND (P18)
BTNL (P17)
BTNR (M17)
CPU RESET
PROG
DONE
MODE
VGA
MONO AUDIO OUT
ETHERNET
USB HOST
POWER
ON
OFF
JXADC
JA
JB
JC
JD
LD(15..0)
SW(15..0)
INTRODU CARACTERE
LIBER/
OCUPAT
RESET