UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Facultatea de Automatică și Calculatoare

Calculatoare și Tehnologia Informației

An I, grupa 30213

APLICAȚIE COD CIFRU PENTRU SECURIZAREA DULAPURILOR

Realizat de:

Iosif Adela

Cluj-Napoca

2022

Cuprins

1. Proiectare

a. Specificație proiect

Să se implementeze o aplicație care permite utilizatorului adăugarea unui cifru din 3 caractere distincte pentru securizarea unui dulap (asemănător dulapurilor folosite la vestiarele de sport, mall etc.). Cerințe funcționale:

1) Un led **LIBER_OCUPAT** va avea funcția de a semnala faptul că dulapul este liber (led stins) sau ocupat (led aprins)
2) Utilizatorul va apăsa un buton **ADAUGĂ_CIFRU** pentru a începe introducerea codului. Un led **INTRODU_CARACTERE** se va aprinde pentru a marca starea
3) Utilizatorul va adăuga pe rând 3 caractere cu ajutorul butoanelor **UP** și **DOWN**
4) Caracterele sunt cuprinse în intervalul 0-1-...-8-9-A-B-..-F
5) Caracterul curent este afișat pe SSD
6) Pentru trecerea la următorul caracter utilizatorul va apăsa butonul **ADAUGA_CIFRU**
7) Caracterul introdus anterior rămâne afișat
8) Următorul caracter este vizibil pe afișaj pe poziția următoare
9) După introducerea celui de-al treilea caracter, la apăsarea butonului **ADAUGA_CIFRU**, afișajul SSD se va stinge, iar cifrul va fi în stare blocată prin aprinderea ledului **LIBER_OCUPAT**.
10) Ledul **INTRODU_CARACTERE** se va stinge
11) Existența unui buton/switch RESET în timpul introducerii cifrului pentru revenire în stare inițială (ledul **LIBER_OCUPAT** se va stinge, afișajul SSD este gol, ledul **INTRODU_CARACTERE** se va stinge)
12) Utilizatorul va apăsa butonul/switch **ADAUGA_CIFRU** pentru a începe introducerea codului pentru deblocarea dulapului
13) Se vor relua pașii 2-8
14) La introducerea ultimului caracter, la apăsarea butonului **ADAUGA_CIFRU** se va face verificarea, dacă codul introdus corespunde cu codul anterior
15) În cazul de egalitate, ledul **LIBER_OCUPAT** se va stinge, ledul **INTRODU_CARACTERE** se va stinge, afișajul SSD se golește
16) În cazul de inegalitate, ledul **LIBER_OCUPAT** va rămâne aprins, ledul **INTRODU_CARACTERE** se va stinge, afișajul SSD se golește.

b. Schema bloc

Am început prin a proiecta cutia neagră a automatului pentru a stabili intrările și ieșirile necesare.

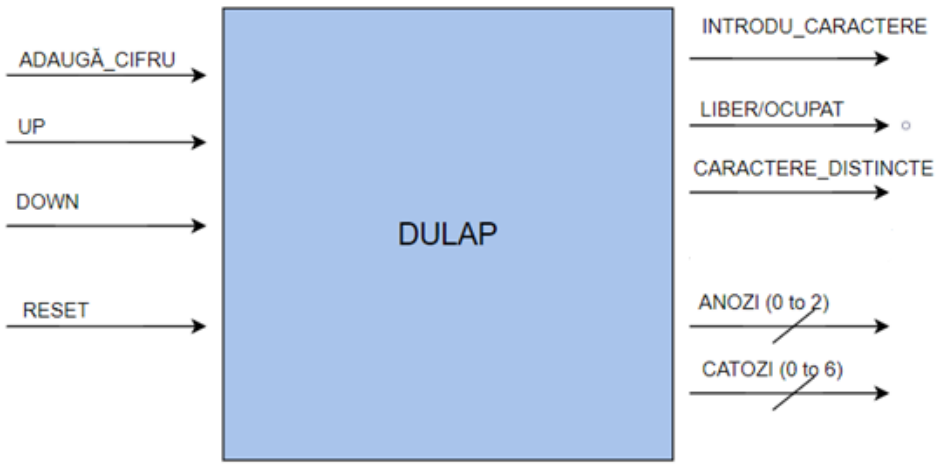
Intrările automatului sunt:

- **ADAUGA_CIFRU**: buton cu care se începe introducerea codului (start), trecerea la caracterul următor și pentru blocarea dulapului
- **BTN_UP**: buton pentru incrementarea numerelor pe afișor
- **BTN_DOWN**: buton pentru decrementarea numerelor pe afișor
- **CLK**: intrare pe un bit (integrată în plăcuță)
- **RESET**: intrare pe 1 bit; se activează când se dorește revenirea în starea inițială

Ieșirile automatului sunt:

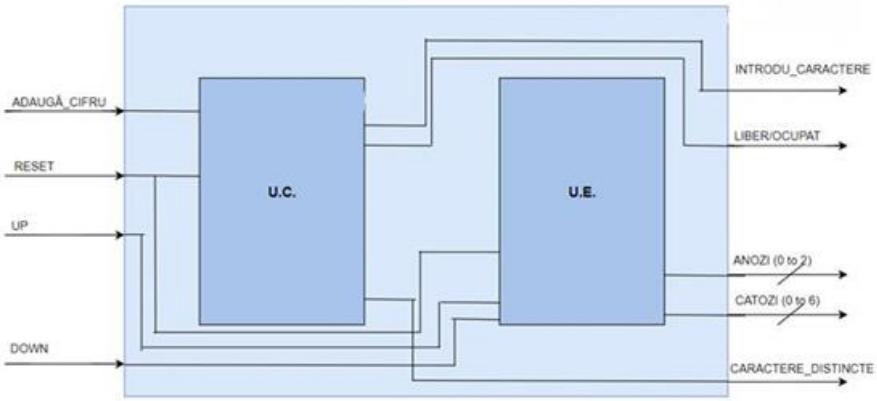
- **ANOZI**: indică care anod este activat (ieșire pe 4 bit)
- **CATOZI**: arată catozii activați (ieșire pe 7 bit)
- **LIBER_OCUPAT**: led care arată utilizatorului dacă dulapul este ocupat (led aprins) sau liber (led stins)
- **INTRODU_CARACTERE**: led care sugerează utilizatorului că automatul este pregătit pentru introducerea cifrului
- **CARACTERE_DISTINCTE**: led care indică dacă caracterele introduse sunt distincte sau nu, ledul este aprins dacă caracterele sunt distincte și stins dacă nu sunt distincte

Cutia neagră a sistemului este următoarea:



c. Unitatea de control și cea de execuție

După stabilirea intrărilor și a ieșirilor am continuat prin a dezvolta cutia neagră și a stabili legăturile dintre componentele acesteia, unitatea de control și cea de execuție.

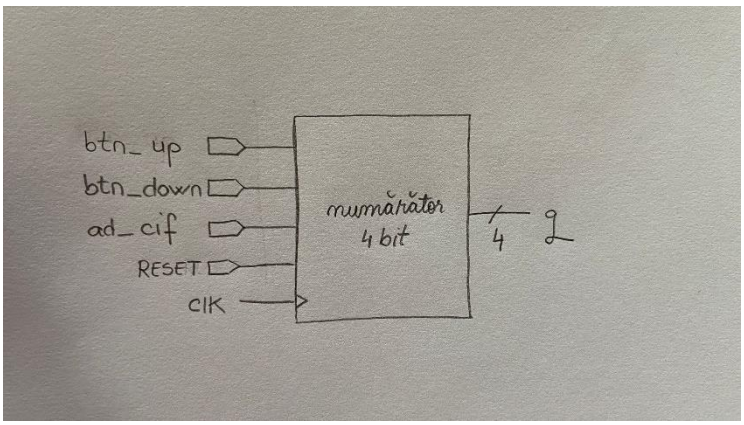


❖ Unitatea de execuție

În unitatea de execuție avem următoarele componente:

- **1 numărător reversibil pe 4 bit**

Folosim un numărător reversibil pe 4 bit care să numere de la 0 la F, cu incrementare sau decrementare la apăsarea butoanelor **BTN_UP** sau **BTN_DOWN** pentru selectarea fiecărui caracter de către utilizator. Acesta (numărătorul) dispune de un buton de **RESET** și un buton **ADAUGA_CIFRU** ambele aducând numărătorul la starea inițială pentru a schimba un caracter, respectiv pentru a seta caracterul următor.




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;


entity numarator is
    Port (clk : in STD_LOGIC;
          btn_up: in std_logic;
          btn_down: in std_logic;
          reset: in std_logic;--apasare buton reset
          ad_cif: in  std_logic;--cand apas adauga_cifre
          q : out STD_LOGIC_VECTOR (3 downto 0)--iesirea care intra in registru
         );
end numarator;
```

```
architecture arh_numarator of numarator is
    signal en_up, en_down, en_adauga_cifru, en_reset: std_logic;

    component  debouncer_afis is
        Port ( button : in STD_LOGIC;
               clk : in STD_LOGIC;
               en : out STD_LOGIC);
    end component ;

    component  bistD is
    Port ( d, clk  : in STD_LOGIC;
           y : out STD_LOGIC);
end component ;

begin
    buton1: debouncer_afis  port map(button => btn_up, clk => clk ,en => en_up);
    buton2 : debouncer_afis  port map(button => btn_down, clk => clk ,en => en_down);
    button3: debouncer_afis port map(button => ad_cif, clk => clk, en => en_adauga_cifru);
    button4: debouncer_afis port map(button => reset, clk => clk, en => en_reset);

    process(clk, en_adauga_cifru, en_up, en_down, en_reset )
        variable count: std_logic_vector(3 downto 0) := "0000";
    begin
        if en_reset = '1' or falling_edge (en_adauga_cifru ) then count :="0000";
        end if;
       if rising_edge (clk)  then
            if en_up = '1' then
                count := count+1;
            end if;
            if en_down = '1' then
                count := count-1;
            end if;
        end if;
        q <= count;
    end process;
end arh_numarator;
```

- **un registru pe 12 bit**

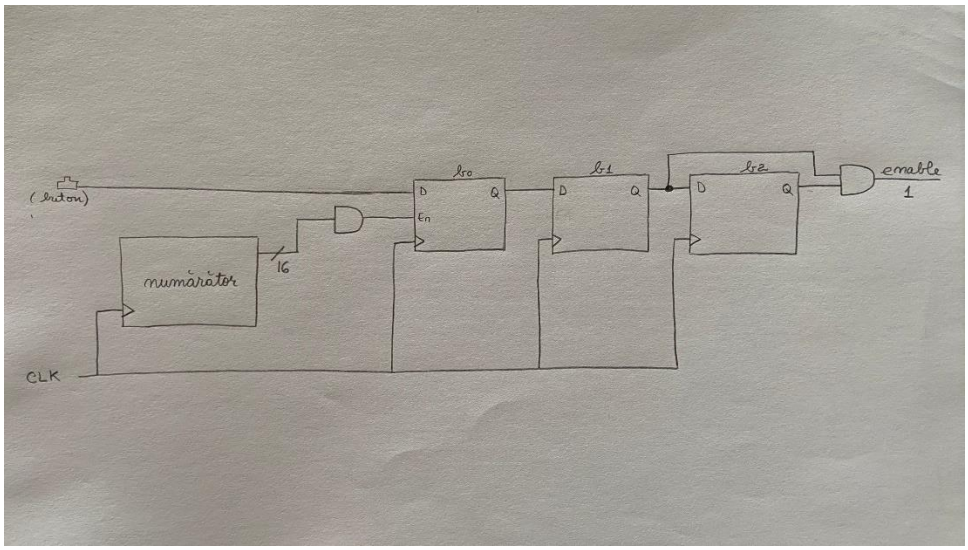
Utilizăm un registru pe 12 bit (care simulează 3 registre pe 4 bit) pentru memorarea codului de către utilizator. Registrul știe pe ce poziție să memoreze caracterul introdus în funcție de numărul de apăsări al butonului **ADAUGA_CIFRU** (transmis în entitatea registru_12B).

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity registru_12B is
    port (clk: in std_logic;
          reset: in std_logic;
          nradcif: inout  std_logic_vector(1 downto 0) ;--numara de cate ori a fost apasat butonul adauga cifru
          d: inout std_logic_vector(3 downto 0);--intrare care vine din iesirea numaratorului
          qout: inout std_logic_vector(11 downto 0)--iesirea de la registru care intra la afisare
         );
end registru_12B;
```

```
architecture arch_registru_12B  of registru_12B is
    component  debouncer_afis is
        Port ( button : in STD_LOGIC;
               clk : in STD_LOGIC;
               en : out STD_LOGIC);
    end component ;
    signal en_reset: std_logic;
begin
    resetare: debouncer_afis port map(button => reset, clk => clk, en => en_reset);
    process (clk, d, nradcif )
        variable aux : std_logic_vector(3 downto 0) := "0000";
    begin
        if(rising_edge(clk)) then
            case nradcif is
                when "01" => qout(11 downto 8) <= d(3 downto 0);
                when "10" => qout(7 downto 4) <= d(3 downto 0);
                when "11" => qout(3 downto 0) <= d(3 downto 0);
                when others =>
            end case;
        end if;
        if en_reset = '1'   then qout <= "000000000000";
        end if;
    end process;

end arch_registru_12B ;
```

- **divizor de frecvență pentru fiecare buton utilizat**

Din cauza uzurii, butoanele au tendința de a trimite semnale multiple la apăsarea lor. Divizorul de frecvență este utilizat ca un filtru care se asigură că la o apăsare de buton este transmis un singur semnal.



```
entity debouncer_afis is
    Port ( button : in STD_LOGIC;
           clk : in STD_LOGIC;
           en : out STD_LOGIC);
end debouncer_afis;

architecture arhit_deb of debouncer_afis is
    --numarator pe 16 biti
    component counter16b
        port(clk : in std_logic ;
             q: out std_logic_vector (15 downto 0));
    end component ;

    --bistabil D
    component bistD
        port(d, clk : in std_logic ;
             y: out std_logic);
    end component ;

    --bistabil D cu enable
    component bistD_en
        port(en, d, clk : in STD_LOGIC;
             y: out std_logic);
    end component ;

    signal q: std_logic_vector (15 downto 0);
    signal y1, y2, y3, s: std_logic ;

  begin

begin
    p1: counter16b port map (clk, q);
    s <= q(15) and q(14) and q(13) and q(12) and q(11) and q(10) and q(9) and q(8) and q(7) and q(6) and q(5)
    and q(4) and q(3) and q(2) and q(1) and q(0);
    p2: bistD_en port map (s, button, clk, y1);
    p3: bistD port map (y1, clk, y2);
    p4: bistD port map (y2, clk, y3);
    en <= y2 and not(y3);
end arhit_deb;
```

- **decodificator**

Utilizăm un decodificator cu intrare pe 4 bit și ieșire pe 7 bit pentru decodificarea fiecărui caracter selectat de utilizator și afișarea SSD, 7-segmente.

	Intrări				Iesiri						
Număr	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0		1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0
10	1	0	1	0	0	0	0	0	0	1	0
11	1	0	1	1	1	1	0	0	0	0	0
12	1	1	0	0	0	1	1	0	0	0	1
13	1	1	0	1	1	0	0	0	0	1	0
14	1	1	1	0	0	1	1	0	0	0	0
15	1	1	1	1	0	1	1	1	0	0	0

❖ Unitatea de control

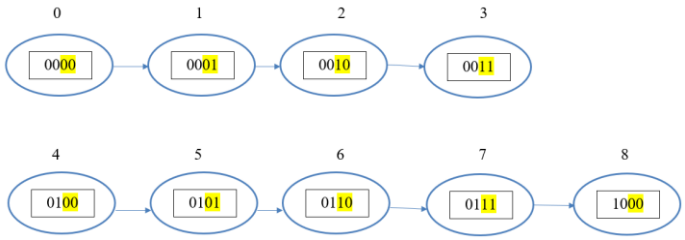
Am recurs la o implementare cablată a unității de control prin utilizarea semnalelor.

```
signal tmp: STD_LOGIC_VECTOR(3 downto 0) := "0000";   --numar de apasari pe butonul adauga_cifru
signal iesire_numarator: STD_LOGIC_VECTOR(3 downto 0);
signal en_adauga_cifru, en_reset: std_logic;
signal caractere :std_logic_vector(11 downto 0);
```

Semnalele *en_adauga_cifru* și *en_reset* sunt port mapate în debouncer pentru a se identifica corect numărul de apăsări ale butoanelor adaugă_cifru, respectiv reset.

```
buton_adcif: debouncer_afis  port map(button => adauga_cifru, clk => clk ,en => en_adauga_cifru);
buton_reset: debouncer_afis  port map(button => reset, clk => clk ,en => en_reset);
```

Semnalul *tmp* reține numărul de apăsări al butonului **ADAUGA_CIFRU**, iar cei mai nesemnificativi 2 bit ai semnalului se transmit registrului (pentru a face selecția poziției pe care se memorează caracterul) și afișorului (pentru a selecta anodul corespunzător). Am recurs la declararea semnalului *tmp* pe 4 bit întrucât cei mai nesemnificativi 2 bit se repetă din 4 în 4.




```
process(clk, en_adauga_cifru, tmp)
    variable nr: std_logic_vector(3 downto 0) := "0000";
begin
    if rising_edge(clk) and en_adauga_cifru = '1' then ---la apasarea butonului sa treaca la anodul urmator
        nr := nr + 1;
    end if;
    if en_reset = '1' then nr := "0000";
    end if;
    tmp <= nr; --0001 0010 0011 0100 0101..
end process;
```

Semnalul *iesire_numarator* reprezintă numărul ales de utilizator, care se modifică în cadrul componentei *numarator*.

```
p1: numarator port map(clk, btn_up, btn_down, reset, adauga_cifru, iesire_numarator);
```

Semnalul *caractere* este transmis ca ieșire pentru registru (pe 12 bit) și afișor.

În funcție de numărul de apăsări al butonului **ADAUGA_CIFRU** caracterele memorate în registru se vor stoca fie în variabila *registru1_blocare* (faza de setare a cifrului – a 4-a apăsare a butonului), fie în variabila *registru2_debl* (faza de încercare de deblocare a dulapului – a 8-a apăsare a butonului).

2. Justificarea soluției alese

Inițial, am desenat o schemă bloc cât mai reprezentativă pentru intrările și ieșirile pe care dorim să le folosim. Am încercat să ilustrăm stările prin care trece automatul, cu ajutorul organigramei, însă am ajuns să abordăm o implementare cablată a unității de control.

Astfel, folosim mai multe componente, care sunt legate în modulul principal afis_buton. Avem o implementare eficientă a soluției, deoarece folosim același numărător pentru toate caracterele introduse și un singur registru pe 12 bit pentru memorare.

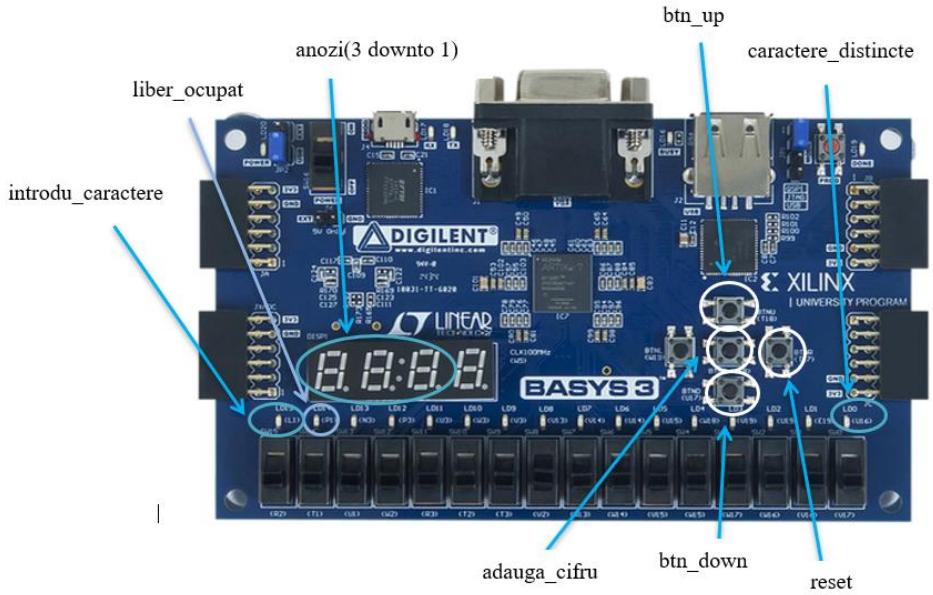
De asemenea, pentru comparare, atât în faza de blocare a dulapului (verificare dacă caracterele sunt distincte), cât și în cea de deblocare (verificare dacă codul introdus corespunde cu cel inițial) am folosit doar câte o condiție care simulează, de fapt, comparatoarele la care ne-am gândit inițial.

```
if caractere(11 downto 8) = caractere(7 downto 4) or caractere(11 downto 8) = caractere(3 downto 0) or caractere(7 downto 4) = caractere(3 downto 0)then
    caractere_distincte <= '0'; --caracterele nu sunt distincte 2 cate 2
else caractere_distincte <= '1';    ---caracterele date sunt distincte
end if;

if registru1_blocare  = registru2_debl   then
    liber_ocupat <= '0';
end if;
```

3. Manual de utilizare și întreținere

Proiectul presupune utilizarea plăcii FPGA Basys 3 și respectiv softul Xilinx Vivado ML Editions 2021.2.

Placa Basys 3



Funcția fiecărei componente:

- pentru semnalul de tact folosim clock-ul integrat în plăcuță (CLK100MHz, respectiv pinul W5)
- leduri:
    - ledul L1 pentru a indica că se poate adăuga cifrul
    - ledul P1 pentru a indica starea dulapului, liber sau ocupat

- ledul U16 pentru a indica dacă caracterele sunt distincte

- butoane:
  - butoanele pentru incrementare/decrementare în momentul adăugării cifrului: *btn_up* = T18, *btn_down* = U17
  - buton **ADAUGA_CIFRU** = U18 pentru a da semnalul de trecere la următorul caracter sau de finalizare a acțiunii de introducere a cifrului
  - buton **RESET** = T17 pentru resetarea registrului
- afișor:
  - anod 1: pin U4
  - anod 2: pin V4
  - anod 3: pin W4

4. Posibilități de dezvoltare

Proiectul ar putea fi dezvoltat prin adăugarea unor funcționalități ca:

- Renunțarea la apăsarea butonului **ADAUGA_CIFRU** după terminarea introducerii fiecărui caracter și integrarea unei perioade predefinite de 6 secunde. Astfel, utilizatorul nu va mai trebui să apese de 3 ori butonul, ci plăcuța va ști după trecerea celor 6 secunde că utilizatorul e pregătit de următorul pas (introducerea următorului număr din cifru sau validarea codului).
- Posibilitatea introducerii unui cod din 3 sau 4 caractere. Dacă butonul număr_caractere nu e apăsat, se rămâne la funcționarea inițială cu un cod din 3 numere, iar dacă butonul se apasă, atunci se va permite folosirea unui cod din 4 numere.
- Adăugarea unui led pentru securitate, care se va aprinde dacă se încearcă deblocarea dulapului de 5 ori cu un cod incorect.

5. Bibliografie

https://www.fpga4student.com (secțiunea VHDL projects), util pentru a înțelege funcționarea afișorului pe plăcuța Basys 3 (Seven-Segment Display)

VHDL Reference Guide