**Universitatea Tehnică "Gheorghe Asachi" din Iaşi**

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

# ELECTRONICĂ DIGITALĂ

# proiect

## Tema: ALU – v4


Studenţi:
**Hriscu Cornelia-Ștefana**
**Leonte Bianca-Florina**
**Pîslariu Andreea**
Grupa : 1210B

Coordonator:
**Asist. Drd. Marius Obreja**


**2022**

# Tema proiectului:

## ALU – v4

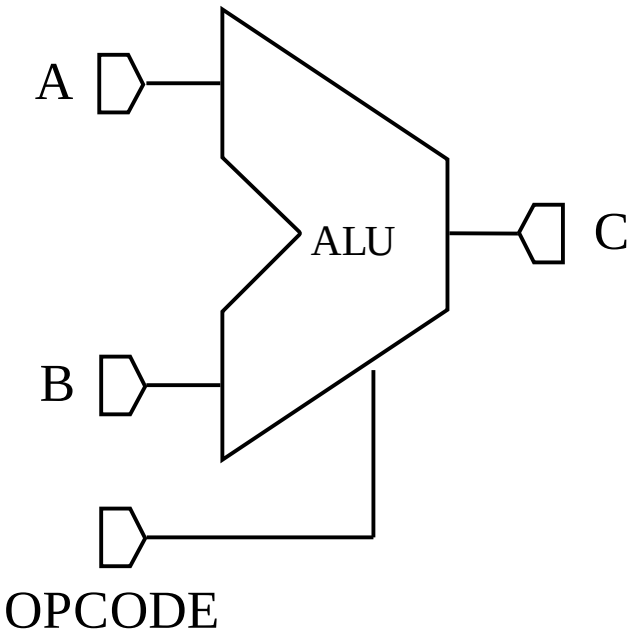
### 1. Specificaţiile proiectului:

Să se implementeze în FPGA prin descriere în limbaj VHDL, utilizând programul VIVADO, modulul prezentat în figura 1 care este descris prin urmatoarele specificaţii:

a) operanzii A şi B au dimensiunea de 8 biţi
b) operaţiile vor fi stabilite prin portul de intrare OPCODE
c) lista de operaţii aritmetice: *, /, +, -

Rezultatele vor fi asignate la portul C şi vor fi vizualizate prin LED-urile de pe placa de dezvoltare.

Descrierea va fi făcută în mod structural.




Fişierul bitstream rezultat în urma procesului de implementare va fi verificat utilizând placa de dezvoltare BASYS3.

# 2. ALU- v4

Modulul ALU_V4 are următoarele funcționalități:

Proiectul conține entitatea ALU-v4 în care avem declarat vectorii si variabilele utilizate. Structura conține ca input 2 intrări de tip std_logic_vector pe 8 biți (A și B) și 5 butoane utilizate la selecția operației aritmetice dorite prin portul de intrare OPCODE: *, /, +, - și reset. Rezultatele vor fi asignate la portul C printr-un vector de ieșire de 8 biți. În efectuarea operațiilor se va ține cont de bitul de Carry care va fi semnalat în cazul unei depășiri.

În arhitectură se scrie codul care realizează operațiile menționate. Se va folosi o buclă if pentru determinarea operației dorite la apăsarea unuia din cele 5 butoane de pe placa Basys3. După efectuarea operației, LED-urile de pe placa de dezvoltare se vor colora pentru valorile de "1" logic și vor rămâne inactive pentru valorile de "0" logic. Numerele pe biți cu care se vor lucra vor fi setate din switch-uri astfel: primele 8 vor corespunde numărului A, iar urmatoarele 8 celui de al doilea număr B. Un LED de pe placă va fi asignat bit-ului de Carry care va fi semnalat în cazul unei depășiri.

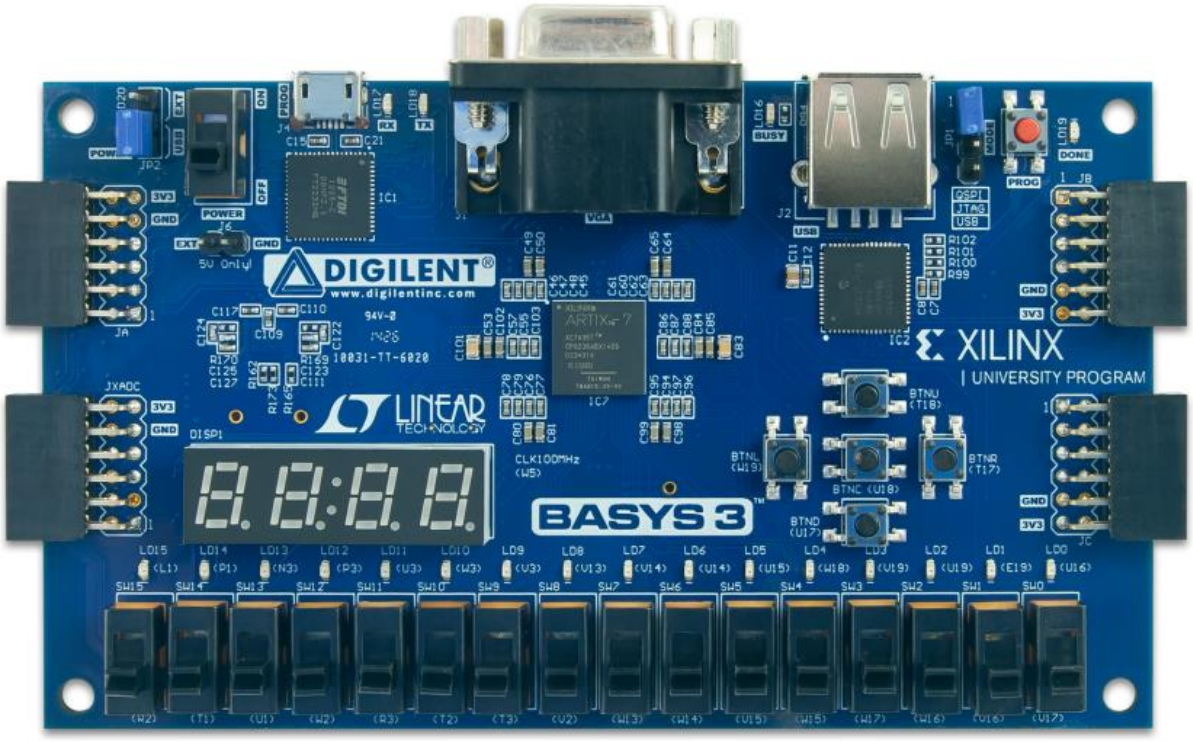



Fig. 2 Placa de dezvoltare Basys 3 - vedere frontală

# 3. Metoda de implementare

Pentru implementarea acestui proiect s-au folosit programul de sinteză Vivado și limbajul VHDL. Implementarea proiectului a fost făcută printr-o descriere comportamentală. S-a proiectat entitatea ALU-v4. În acest scop s-au avut în vedere următoarele aspecte :

- Librariile utilizate in implementare sunt: IEEE.STD_LOGIC_1164.ALL, IEEE.STD_LOGIC_UNSIGNED.ALL; IEEE.NUMERIC_STD.all;
- Operandul A este pe 8 biți și este dat de switch-urile: W13, W14, V15, W15, W17, W16, V16, V17 2.
- Operandul B este pe 8 biți și este dat de switch-urile: R2, T1, U1, W2, R3, T2, T3, U2
- Selectarea operatiei prin apăsarea unuia dintre cele 5 butoane: BTNU(T18), BTNR(T17), BTND(V17), BTNL(W19), BTNC(V18)
- Utilizarea unui variabile locale tmp în care s-a calculat rezultat operatiilor, asignat apoi operatorului C
- Ieșirea asignata la portul C este conectată la LED-urile plăcii de dezvoltare BASYS3 (operandul C(ALU_Result) este compus din LED-urile V14, U14, U15, W18, V19, U19, E19, U16)
- S-a ținut cont de bitul de Carry Flag în cazul unei depășiri fiind semnalat prin urmatorul LED de lânga rezultatul operației
- Pentru a corela entitatea ALU_V4 cu placa de testare s-a creat fișierul de constrângeri . În acesta s-au stabilit legăturile dintre operatorii de intrare( A , B , btnC, btnU, btnD ,btnL, btnR) și switch-uri, respectiv butoane. Operatorii ALU_result și CarryOut au fost corelați cu led-urile.

Fișierul bitstream creat de programul Vivado a fost testat cu ajutorul plăcii BASYS 3 Artix-7 xc7a35tcpg236-1.

# 4. Descrierea (scurtă) a sistemului de dezvoltare BASYS 3

Placa Basys3 este o platformă de dezvoltare digitală completă, gata de utilizare, bazată pe ultima tehnologie Artix-7 Array Gate Gate Programable (FPGA) de pe Xilinx. Cu FPGA de mare capacitate, cost general redus și colectarea de porturi USB, VGA și alte porturi, Basys3 poate găzdui modele care variază de la circuite combinaționale introductive pentru circuite secvențiale complexe, cum ar fi procesoare și controlere încorporate.

Aceasta include suficiente întrerupătoare, LED-uri și alte dispozitive I / O pentru a permite finalizarea unui număr mare de proiecte fără necesitatea oricărui hardware suplimentar și destui pini de I / O FPGA pentru a permite extinderea proiectelor folosind Pmod-uri Digilent sau alte plăci și circuite personalizate. Artix-7 FPGA este optimizat pentru logica de înaltă performanță și oferă mai multă capacitate, performanțe superioare și multe altele resurse decât proiectele anterioare .

Basys3 funcționează cu noua serie de design Vivado de înaltă performanță a Xilinx. Vivado include multe instrumente noi și fluxuri de proiectare care facilitează și îmbunătățesc cele mai recente metode de proiectare. Rulează mai repede, permite utilizarea mai bună a FPGA resurse și permite proiectanților să-și concentreze timpul evaluând alternativele de proiectare.

# 5. Editarea fişierului VHDL

----------ALU_v4.vhd(TOP MODULE)----------

## Entitatea ALU_v4:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.NUMERIC_STD.all;

entity ALU is
    Port (
    A, B      : in  STD_LOGIC_VECTOR(7 downto 0);
    ALU_Out   : out  STD_LOGIC_VECTOR(8 downto 0);
    Carryout : out std_logic;        -- Carryout flag
    btnC : in std_logic; --Reset
    btnU: in std_logic; --Addition
    btnD : in std_logic; --Subtraction
    btnR: in std_logic; --Multiplication
    btnL : in std_logic --Division
    );
end ALU;

architecture Behavioral of ALU is

signal ALU_Result : std_logic_vector (8 downto 0);
signal tmp: std_logic_vector (8 downto 0);

begin
    process(A,B, btnC, btnD, btnL, btnR, btnU)
```

## If statement adunare:

```
begin

--Addition

    if btnU='1' then
        tmp <= std_logic_vector(to_unsigned((to_integer(unsigned(A)) + to_integer(unsigned(B))),9)) ;
        ALU_Result <= tmp(8 downto 0);
    end if;
```

## If statement scădere:

```
--Subtraction

   if btnD='1' then
       if A>B then
            tmp <= std_logic_vector(to_unsigned((to_integer(unsigned(A)) - to_integer(unsigned(B))),9)) ;
            ALU_Result <= tmp(8 downto 0);
        else
            tmp <='0'& A;
            ALU_Result <= tmp(8 downto 0);
        end if;
   end if;
```

## If statement înmulțire:

```
    --Multiplication

      if btnR='1' then
          tmp <= std_logic_vector(to_unsigned((to_integer(unsigned(A)) * to_integer(unsigned(B))),9)) ;
          ALU_Result <= tmp(8 downto 0);
      end if;
```

## If statement împărțire:

```
   --Division

     if btnL='1' then
         tmp <= std_logic_vector(to_unsigned(to_integer(unsigned(A)) / to_integer(unsigned(B)),9)) ;
         ALU_Result <= tmp(8 downto 0);
      end if;
```

## If statement reset:

```
  --Reset

    if btnC='1' then
        tmp <= ('0' & A) + ('0' & B);
    end if;
```

## Asignare rezultat final:

```
    end process;
    ALU_Out <= ALU_Result; -- ALU out
    Carryout <= tmp(8); -- Carryout flag
end behavioral;
```

Pentru creșterea lizibilității codului, acesta a fost decupat din fișierele vhd văzute prin intermediul programului Notepad++.

## 6. Editarea fişierului de constrângeri

### Switch-uri:

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {A[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]
set_property PACKAGE_PIN V16 [get_ports {A[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]
set_property PACKAGE_PIN W16 [get_ports {A[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]
set_property PACKAGE_PIN W17 [get_ports {A[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]
set_property PACKAGE_PIN W15 [get_ports {A[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[4]}]
set_property PACKAGE_PIN V15 [get_ports {A[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[5]}]
set_property PACKAGE_PIN W14 [get_ports {A[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[6]}]
set_property PACKAGE_PIN W13 [get_ports {A[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A[7]}]
set_property PACKAGE_PIN V2 [get_ports {B[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property PACKAGE_PIN T3 [get_ports {B[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property PACKAGE_PIN T2 [get_ports {B[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[2]}]
set_property PACKAGE_PIN R3 [get_ports {B[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[3]}]
set_property PACKAGE_PIN W2 [get_ports {B[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[4]}]
set_property PACKAGE_PIN U1 [get_ports {B[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[5]}]
set_property PACKAGE_PIN T1 [get_ports {B[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[6]}]
set_property PACKAGE_PIN R2 [get_ports {B[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B[7]}]
```

### LED-uri:

```
## LEDs
set_property PACKAGE_PIN U16 [get_ports {ALU_Out[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[0]}]
set_property PACKAGE_PIN E19 [get_ports {ALU_Out[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[1]}]
set_property PACKAGE_PIN U19 [get_ports {ALU_Out[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[2]}]
set_property PACKAGE_PIN V19 [get_ports {ALU_Out[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[3]}]
set_property PACKAGE_PIN W18 [get_ports {ALU_Out[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[4]}]
set_property PACKAGE_PIN U15 [get_ports {ALU_Out[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[5]}]
set_property PACKAGE_PIN U14 [get_ports {ALU_Out[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[6]}]
set_property PACKAGE_PIN V14 [get_ports {ALU_Out[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[7]}]
set_property PACKAGE_PIN V13 [get_ports {ALU_Out[8]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {ALU_Out[8]}]
set_property PACKAGE_PIN V3 [get_ports {Carryout}]
    set_property IOSTANDARD LVCMOS33 [get_ports {Carryout}]
```

### Butoane externe:

```
##Buttons
set_property PACKAGE_PIN U18 [get_ports btnC]
    set_property IOSTANDARD LVCMOS33 [get_ports btnC]
set_property PACKAGE_PIN T18 [get_ports btnU]
    set_property IOSTANDARD LVCMOS33 [get_ports btnU]
set_property PACKAGE_PIN W19 [get_ports btnL]
    set_property IOSTANDARD LVCMOS33 [get_ports btnL]
set_property PACKAGE_PIN T17 [get_ports btnR]
    set_property IOSTANDARD LVCMOS33 [get_ports btnR]
set_property PACKAGE_PIN U17 [get_ports btnD]
    set_property IOSTANDARD LVCMOS33 [get_ports btnD]
```

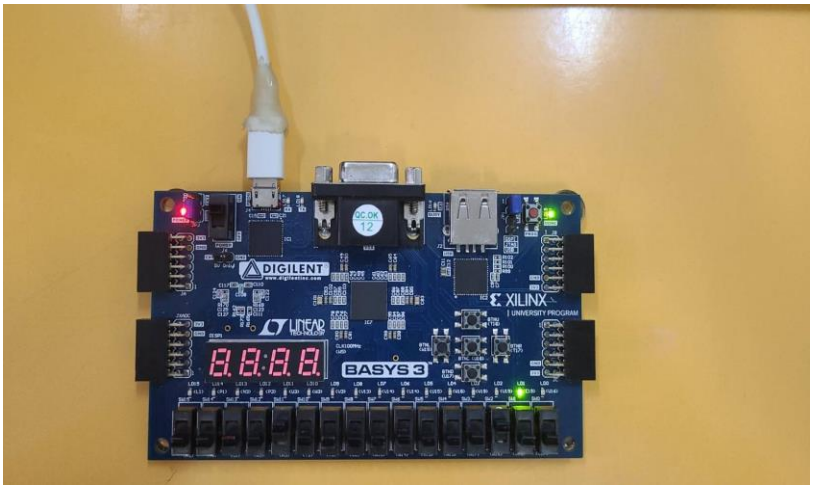
# 7. Descrierea paşilor de sinteză şi testarea circuitului rezultat

1. S-a creat un proiect nou în programul Vivado
2. S-a implementat modulul “ALU_V4” printr-o descriere comportamentală având drept “procese”o serie de operații aritmetice: adunare, scădere, înmulțire, împărțire
3. S-a editat fișierul de constrângeri în vederea realizării legăturilor între cele 16 switch-uri, cele 5 butoane care au rolul de a alege operația dorită și ieșirea reprezentată de cele 8 led-uri și al 9-lea led pentru semnalarea bitului de Carry în cazul unei depășiri
4. S-a realizat analiza RTL(Register Transfer Level)
5. S-a sintetizat modulul(pentru se vedea design-ul sintetizat)
6. S-a lansat implementarea proiectului care a avut ca efect final generarea fișierului bitstream
7. S-a programat placa de dezvoltare BASYS 3 cu fișierul bitstream și s-a testat funcționarea corespunzătoare a modulului implementat

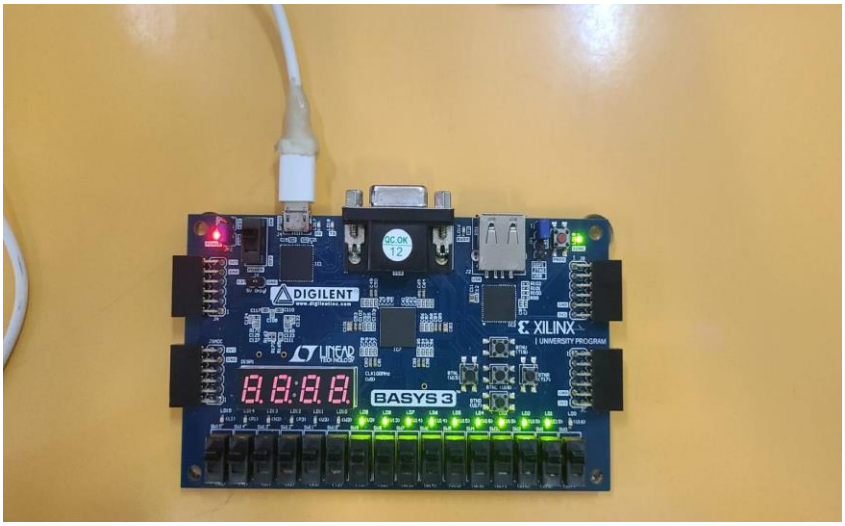
# 8. Fotografii cu funcționarea proiectului:

## Adunare

S-a realizat operația de adunare prin apăsarea butonului btnU(T18) între numerele A=00000001, respectiv B=00000001 rezultatul fiind 00000010 cu led-urile aprinse corespunzător.

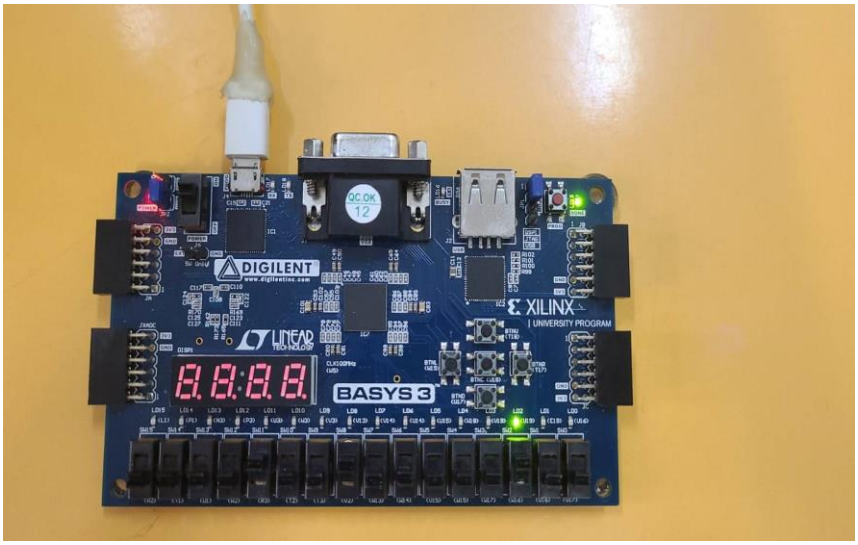


S-a realizat operația de adunare prin apăsarea butonului btnU(T18) între numerele A=11111111, respectiv B=11111111 rezultatul fiind  111111110, cu led-urile aprinse corespunzător (se observă semnalarea bit-ului de Carry datorită depășirii).




## Scădere

S-a realizat operația de scădere prin apăsarea butonului btnD(V17) între numerele A=00001001, respectiv B=00000101 rezultatul fiind  00000100, cu led-urile aprinse corespunzător.

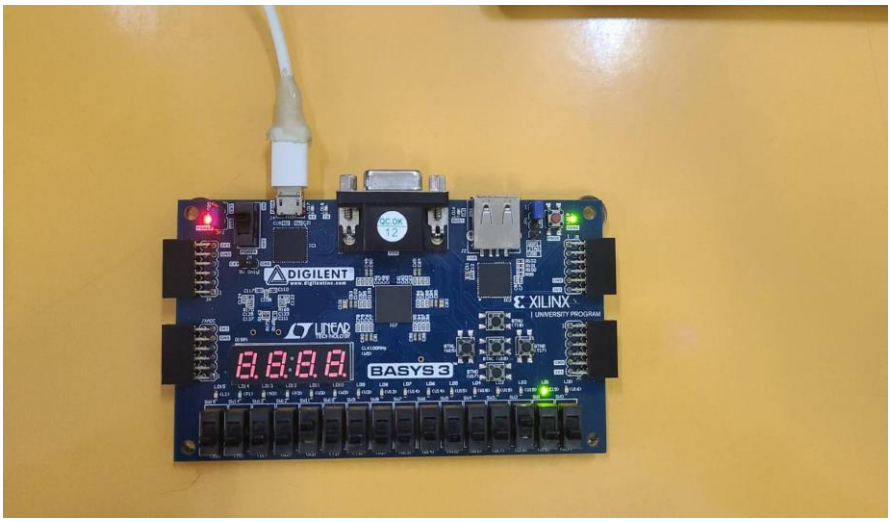


## Înmulțire

S-a realizat operația de înmulțire prin apăsarea butonului btnR(T17) între numerele A=00000011, respectiv B=00000010 rezultatul fiind  00000110, cu led-urile aprinse corespunzător.


## Împărțire

S-a realizat operația de împărțire prin apăsarea butonului btnL(W19) între numerele A=00001000, respectiv B=00000100 rezultatul fiind  00000010, cu led-urile aprinse corespunzător.



# 9. Concluzii

În concluzie, s-a implementat un ALU cu operații aritmetice (adunare, scădere, înmulțire, împărțire) între două numere A și B pe 8 biți fiecare, cu posibilitatea de alegere a uneia dintre acestea prin apăsarea unuia din cele 4 butoane de pe placa de testare, cel de al 5-lea buton fiind pentru resetarea și afișarea rezultatelor prin intermediul led-urilor.

# Bibliografie:

1. VHDL Reference Manual,
http://www.ics.uci.edu/~jmoorkan/vhdlref/Synario%20VHDL%20Manual.pdf

2. BASYS 3 Reference Manual, https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manua