Universitatea Tehnică "Gheorghe Asachi" din Iaşi

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

ELECTRONICĂ DIGITALĂ

Proiect

Tema: **ALU – v2**

Student:
Petcu Andrei Dănuț

Grupa : 1212B

Coordonator:
Asistent doctorand Ionica Pletea

2023

Tema proiectului:

ALU – v2

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:

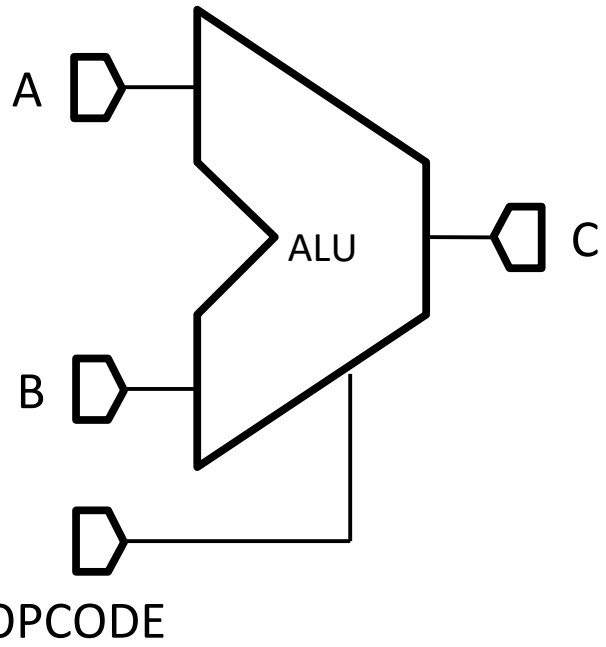
a) operanzii A şi B au dimensiunea de 2 biţi

b) operaţiile vor fi stabilite prin portul de intrare OPCODE

c) lista de operaţii: +, - , *, /

d) afisarea se va face pe displayul 7 segmente

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



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

2. Modulul ALU – v2

Modulul ALU – v2 are următoarele funcționalități:

Unitatea aritmetică logică, ALU este un circuit electronic digital complex care poate efectua operații aritmetice și logice. În diagramele-bloc de computere, unitatea aritmetică logică este reprezentată ca un modul funcțional, componentă a schemei de principiu a unui calculator electronic. Unitatea logică aritmetică este utilizată pentru a efectua transformări logice și aritmetice pe operanții necesari, adesea comenzi sau coduri de numere.

Sarcina principală a ALU este de a procesa datele stocate în memoria RAM a computerului. În plus, o unitate logică aritmetică este capabilă să producă semnale de control care direcționează un calculator să selecteze calea corectă pentru a efectua procesul de calcul necesar, în funcție de tipurile de date rezultate. Toate operațiunile implică circuite electronice, fiecare dintre ele fiind structurate în mii de elemente.

În funcție de semnalele introduse, unitățile ALU execută diferite tipuri de operații cu două numere. Orice unitate logică aritmetică a calculatorului asigură implementarea a patru acțiuni de bază, schimburi, precum și transformări logice. Setul de operațiuni ALU este principala sa caracteristică.

Pentru realizarea funcțiilor sale, unitatea aritmetică şi logică utilizează câteva registre proprii speciale:

1. Acumulatorul - registru de uz general care este utilizat de ALU pentru stocarea unuia dintre operanzi şi pentru rezultatul operației;

2. Registrul F - este registrul fanioanelor de condiții (Flags) şi conține celule de memorie independente, cu funcții specifice, pentru înregistrarea unor informații ce rezultă din operațiile aritmetice şi logice (semnul rezultatului, paritatea, existența bitului de transport sau împrumut, depăşirea domeniului şi altele);

3. Registrul de deplasare - utilizat pentru deplasări spre stânga sau spre dreapta a unui operand.

3. Metoda de implementare

Pentru realizarea proiectului se vor folosi resursele**: limbajul VHDL, circuit FPGA, programul de sinteză Vivado.**

VHDL (abrevierea VHSIC HDL) este acronimul folosit pentru Very High SpeedIntegrated Circuit Hardware Description Language.

Este vorba despre un limbaj de descriere a hardware-ului , destinat descrierii comportamentuluișsi/sau arhitecturii unui "modul" electronic logic, cu alte cuvinte al unei funcțiuni logice combinatorii și/sau secvențiale. E una din uneltele principale pentru proiectarea circuitelor integrate moderne, aplicat cu succes in campul microprocesoarelor (DSP, acceleratoare grafice), în telecomunicatii (TV, celulare), automobile (navigație, sisteme de control al stabilității) și altele.

Un **FPGA** (Field Programmable Gate Array) este un circuit integrat digital configurabil, de către utilizator, după ce a fost fabricat (spre deosebire de dispozivele a căror funcție este implementată in procesul de fabricație). Configurarea FPGA se face, în general, cu ajutorul unui limbaj de descriere hardware HDL, similar cu cel folosit pentru dispozivele ASIC , dezvoltându-se recent și compilatoare care traduc instrucțiuni din limbajul C în limbaje HDL. FPGA-urile sunt alcătuite din blocuri logice configurabile (programabile) legate între ele de o serie de conexiuni configurabile la rândul lor.

Vivado Design Suite este o gamă software produsă de Xilinx pentru sinteza și analiza proiectelor HDL, înlocuind Xilinx ISE cu caracteristicisuplimentare pentru dezvoltarea sistemelor și sinteza la nivel înalt. Vivado reprezintă o rescriere fundamentală și o gândire completă a întregului flux de proiectare (comparativ cu ISE).

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

Placa de dezvoltare BASYS 3 este un circuit de dezvoltare complet și ready-to-use bazat pe ultimele Artix-7 Field Programmable Gate Array(FPGA) produse de Xilinx. Cu o mare capacitate de FPGA și cu o colecție de porturi USB, VGA și altele, placa de dezvoltare BASYS 3 permite proiectarea unor design-uri variate, atât circuite introductorii combinaționale, cât și circuite secvențiale complexe ca procesoarele și controllerele embedded.

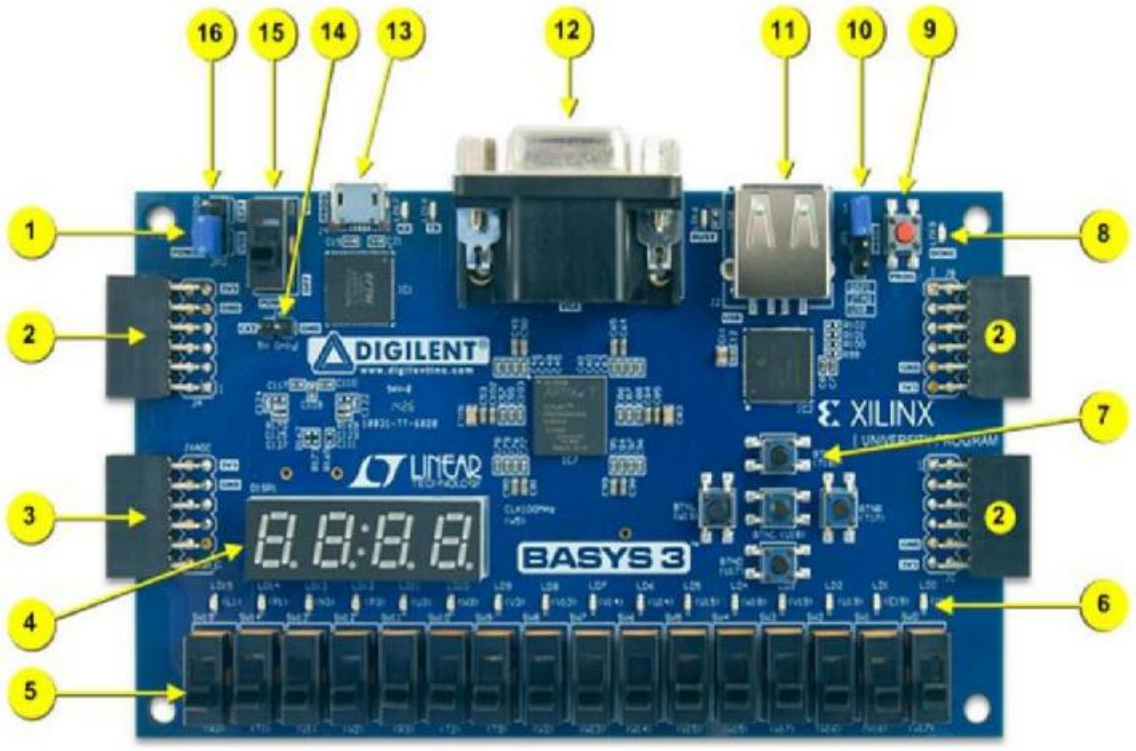


Figure 1. Basys3 board features

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

5. Editarea fişierului VHDL

```
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;
entity Proiect is
    Port ( A : in STD_LOGIC_VECTOR(1 downto 0);
           B : in STD_LOGIC_VECTOR(1 downto 0);
           reset: in boolean;
           operatie : in STD_LOGIC_VECTOR(2 downto 0);
           segment : out STD_LOGIC_VECTOR(6 downto 0));
end Proiect;


architecture Behavioral of Proiect is
    signal C : STD_LOGIC_VECTOR(3 downto 0);
begin
    process(A, B, operatie,reset)
    begin
        case operatie is
        when "000" =>
              C <= ('0' &'0' & A) + ('0' &'0' & B);  -- Adunare
          when "001" =>
```

```
            C <= ('0' &'0' & A) - ('0' &'0' & B);  -- Scadere
        when "010" =>
            C <= (A) * (B);  -- Inmultire
        when "011" =>
            if B /= "00" then
                C<=STD_LOGIC_VECTOR(TO_UNSIGNED((TO_INTEGER(UNSIGNED('0' &'0' & A)) /
TO_INTEGER(UNSIGNED('0' &'0' & B))), 4)); -- Impartire
            else
                C <= "0000";  -- Seteaza la zero in caz de impartire la zero
            end if;
        when others =>
            C <= "0000";
    end case;
  end process;
  process(C)
  begin
    case C is
        when "0000" =>
            segment <= "1000000";  -- 0
        when "0001" =>
            segment <= "1111001";  -- 1
        when "0010" =>
            segment <= "0100100";  -- 2
        when "0011" =>
```

```
            segment <= "0110000";  -- 3
        when "0100" =>
            segment <= "0011001";  -- 4
        when "0101" =>
            segment <= "0010010";  -- 5
        when "0110" =>
            segment <= "0000010";  -- 6
        when "0111" =>
            segment <= "1111000";  -- 7
        when "1000" =>
            segment <= "0000000";  -- 8
        when "1001" =>
            segment <= "0010000";  -- 9
        when others =>
            segment <= "1111111";  -- Afiseaza ceva pentru alte rezultate
    end case;


    case reset is
    when TRUE=>
    segment<="1111111";
    when FALSE=>
    end case;
end process;
```

```
end Behavioral;
```

6. Editarea fişierului de constrângeri

```
##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 {B[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {B[0]}]
set_property PACKAGE_PIN W17 [get_ports {B[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {B[1]}]
set_property PACKAGE_PIN W15 [get_ports {operatie[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {operatie[0]}]
set_property PACKAGE_PIN V15 [get_ports {operatie[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {operatie[1]}]
set_property PACKAGE_PIN W14 [get_ports {operatie[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {operatie[2]}]
set_property PACKAGE_PIN W13 [get_ports {reset}]
        set_property IOSTANDARD LVCMOS33 [get_ports {reset}]


##7 segment display
set_property PACKAGE_PIN W7 [get_ports {segment[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[0]}]
set_property PACKAGE_PIN W6 [get_ports {segment[1]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[1]}]
set_property PACKAGE_PIN U8 [get_ports {segment[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[2]}]
set_property PACKAGE_PIN V8 [get_ports {segment[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[3]}]
set_property PACKAGE_PIN U5 [get_ports {segment[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[4]}]
set_property PACKAGE_PIN V5 [get_ports {segment[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[5]}]
set_property PACKAGE_PIN U7 [get_ports {segment[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {segment[6]}]
```

7. Descrierea paşilor de sinteză şi testarea circuitului rezultat

Pentru rezolarea problemei am definit o entitate cu numele “Proiect”, in care am declarant urmatoarele variabile:

A-Vector pe 2 biti, primul operand;

B-Vector pe 2 biti, al doilea operand;

Reset

Operatie-Vector pe 3 biti, stabilirea operatiei;

Segment-Vector pe 7 biti, pentru afisarea pe display-ul 7 segmente;

C-Vector pe 4 biti, rezulatul operatiilor;

Pentru realizarea temei am folosit mediul de lucru Vivado, creand un nou proiect . Am selectat circuitul FPGA “xc7a35tcpg236-1” pentru a putea incarca designul nostru pe placuța Basys 3. Am implementat schema bloc Proiect, urmata de sinteza , implementarea designului și generarea bitstreamului.

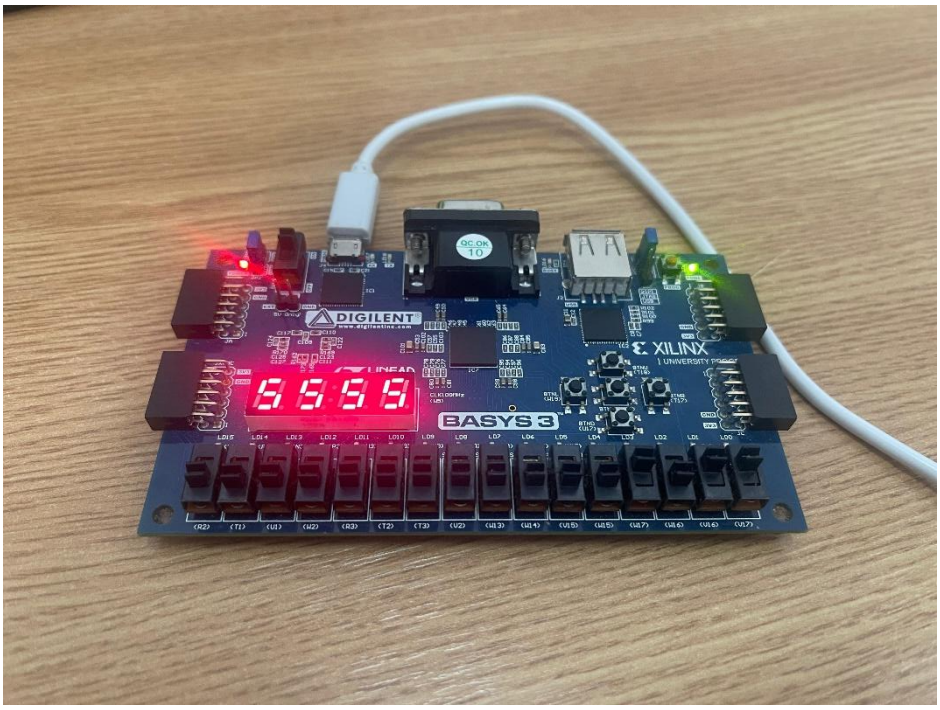
Operatia “+”:

Primul numar (A): 11

Al doilea numar (B): 10

Operatia: 000

Rezultat: (2+3)=5



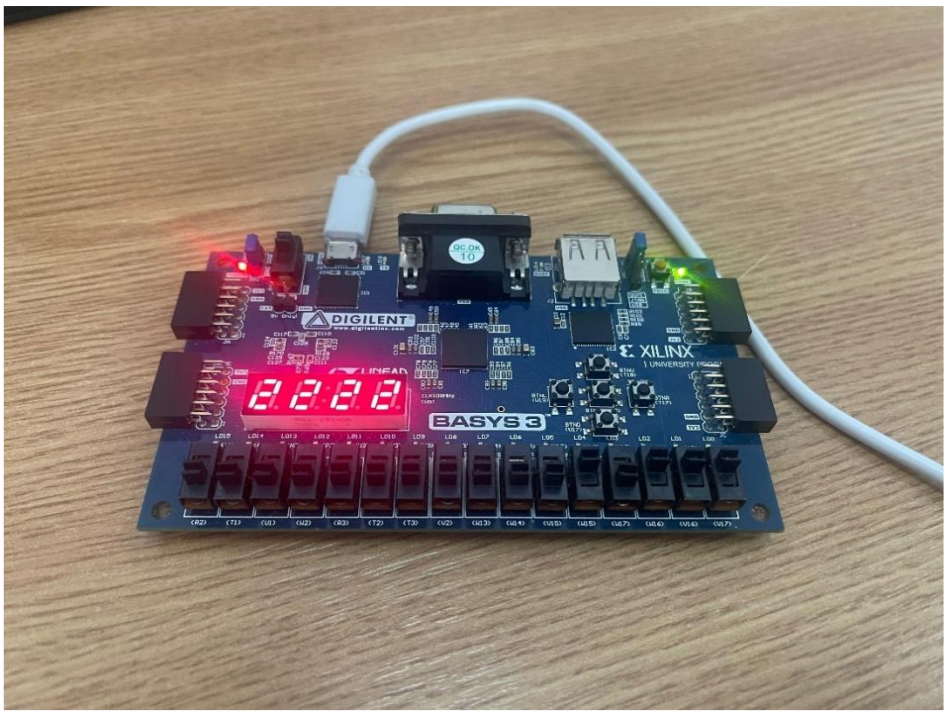
Operatia “-”:

Primul numar (A): 11

Al doilea numar (B): 01

Operatia: 001

Rezultat: (3-1)=2



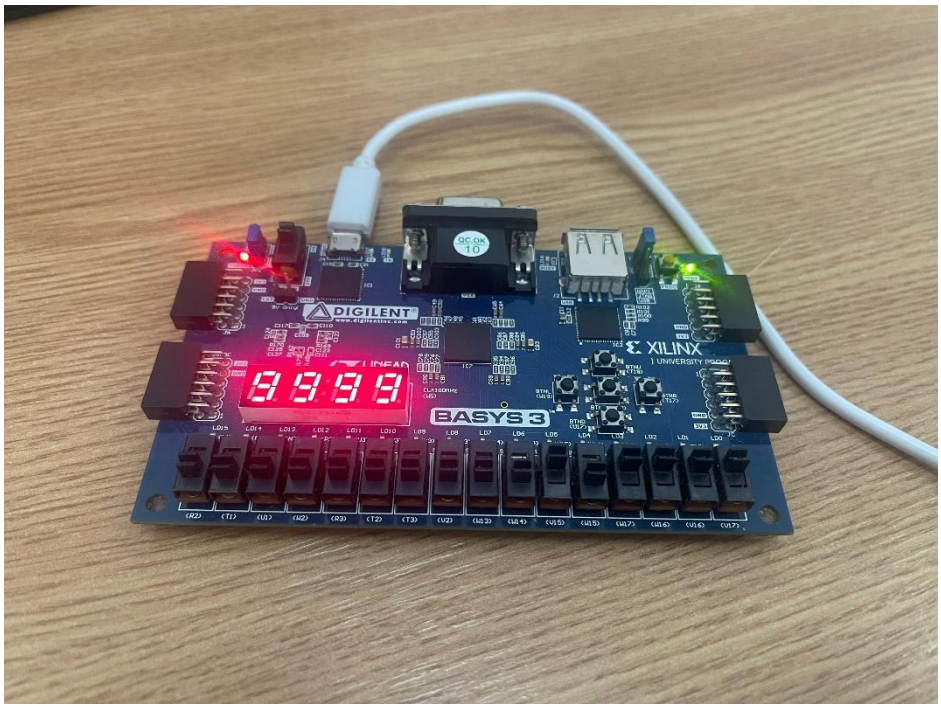
Operatia “*”:

Primul numar (A): 11

Al doilea numar (B): 11

Operatia: 010

Rezultat: (3*3)=9



Operatia “/”:

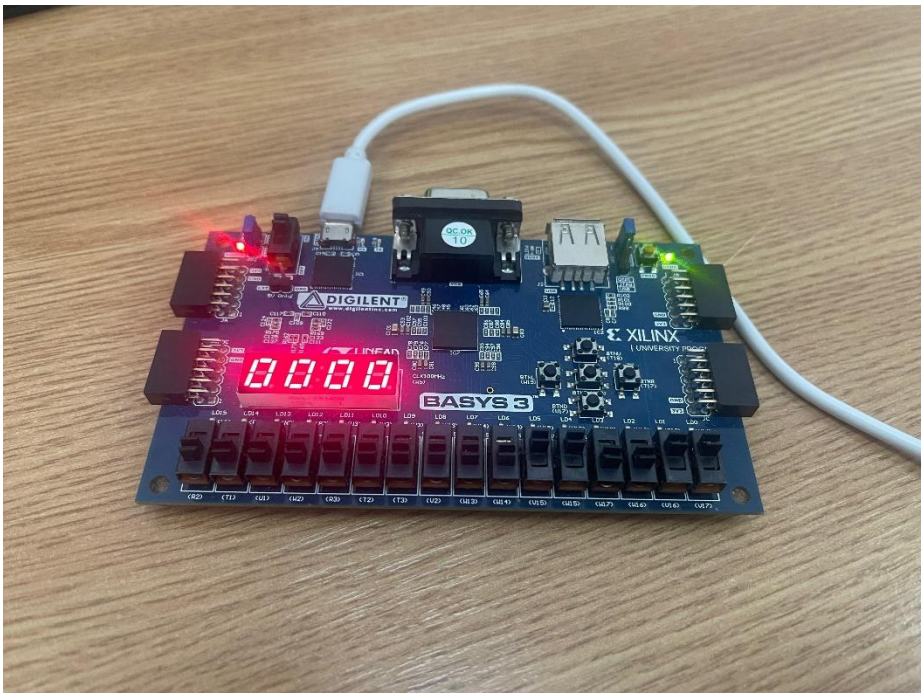
A)

Primul numar (A): 11

Al doilea numar (B): 00

Operatia: 011

Rezultat: EROARE (3/0)



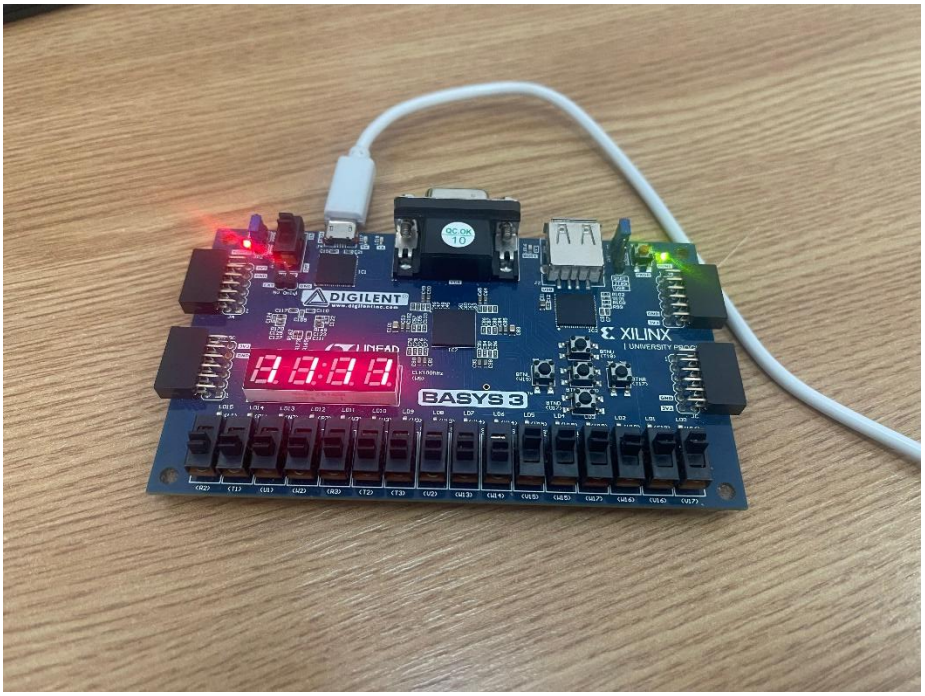
B)

Primul numar (A): 11

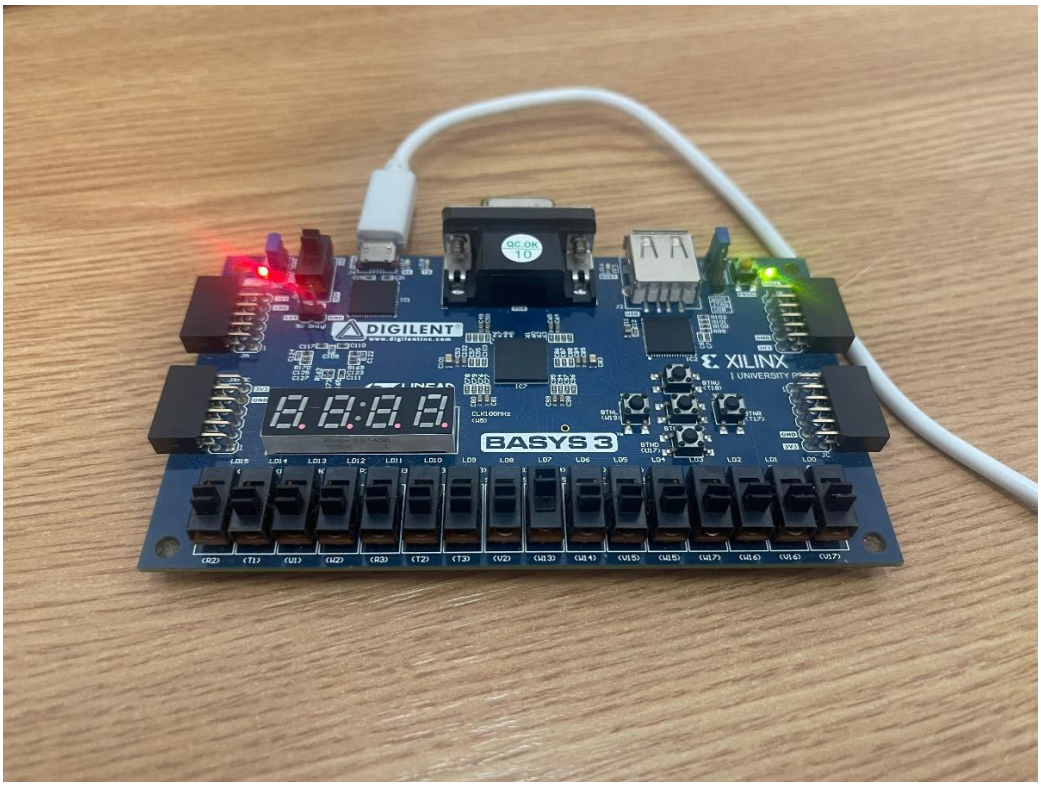
Al doilea numar (B): 11

Operatia: 011

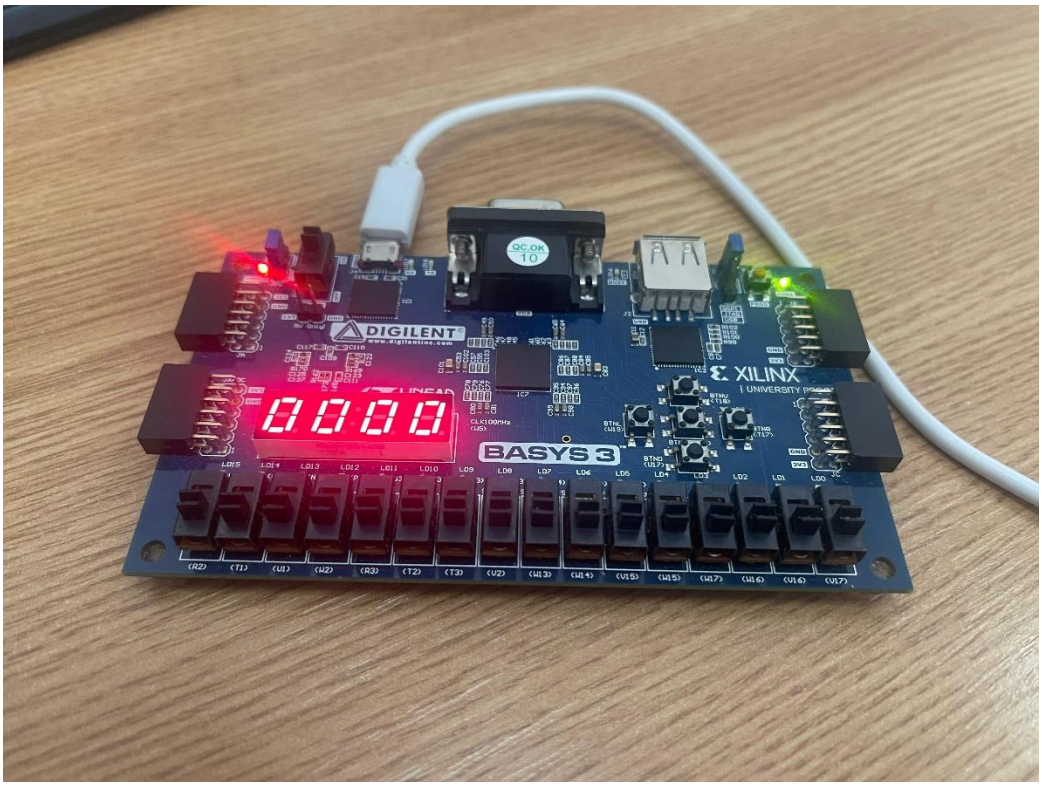
Rezultat: (3/3)=1



Eroare: ON



Eroare: OFF



8. Concluzii

In concluzie, placa de dezvoltare Basys3 reprezinta un instrument puternic si capabil pentru testarea si invatarea facilitatilor limbajului hardware descriptiv VHDL, acesta fiind unul dintre cele mai folosite.

Bibliografie:

1. VHDL Reference Manual, http://www.ics.uci.edu/~jmoorkan/vhdlref/Synario%20VHDL%20Manual.pdf
2. BASYS 3 Reference Manual, Basys 3 Reference Manual - Digilent Reference
3. Trouble on understanding ALU 2-bit design - Electrical Engineering Stack Exchange
4. VHDL - Wikipedia