**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:

* 1. operanzii A şi B au dimensiunea de 2 biţi
  2. operaţiile vor fi stabilite prin portul de intrare OPCODE
  3. lista de operaţii: +, - , \*, /
  4. 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.

ALU

A

B

C

OPCODE

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

1. **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.

## 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.

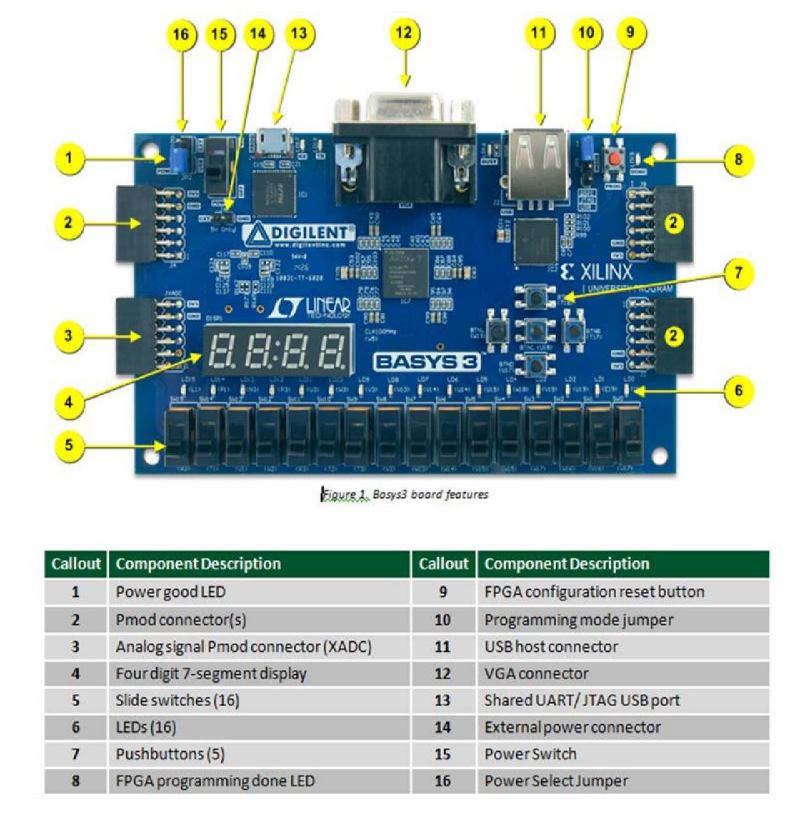
Este vorba despre un limbaj de descriere a hardware-ului , destinat descrierii comportamentuluiși/sau arhitecturii unui "modul" electronic logic, cu alte cuvinte al unei funcțiuni logice combinatorii și/sau secvențiale. E una din uneltele principale pentru proiectarea circuitelor integrate moderne, aplicat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

1. **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.



1. **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;

1. **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]}]

1. **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 “/”:**

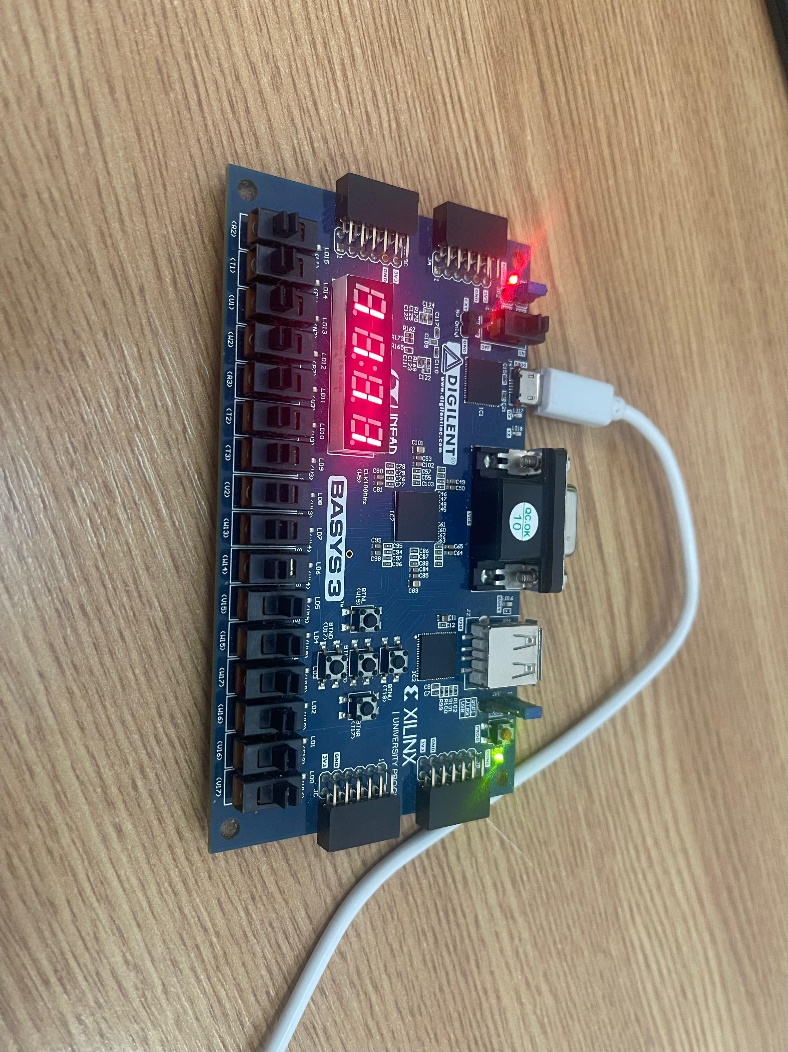
**A)**

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](https://digilent.com/reference/programmable-logic/basys-3/reference-manual?redirect=1)
3. [Trouble on understanding ALU 2-bit design - Electrical Engineering Stack Exchange](https://electronics.stackexchange.com/questions/78351/trouble-on-understanding-alu-2-bit-design)
4. [VHDL - Wikipedia](https://en.wikipedia.org/wiki/VHDL)