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

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

# ELECTRONICĂ DIGITALĂ

# --proiect--

## Tema: CEAS DIGITAL

Student: Toma Anamaria-Ramona
Grupa : 1210B

Coordonator:
Asist. Drd. Ionica Pletea

**2023**

# CEAS DIGITAL

## 1.Specificațiile proiectului

Să se implementeze în FPGA prin descriere în limbaj VHDL, utilizând programul VIVADO, un modul care să determine în timp real minutele și secundele (echivalent unui ceas digital).


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

## 2. Modulul

Pentru a crea ceasul digital am ținut cont de faptul că plăcuța FPGA are un clock intern cu o frecvență de 10 MHz și afișarea se face pe un display cu 7 segmente.

Date de intrare:

clk – de tip STD_LOGIC reprezintă ceasul extern al plăcuței ce are o frecvență de 10 MHz

reset – de tip STD_LOGIC reprezintă un switch de pe plăcuță cu ajutorul căruia vom reseta ceasul digital creat ( adică il vom pune pe 00:00)

Date de intrare-ieșire:

an - un vector cu 4 biți de tipul STD_LOGIC (3 downto 0) cu ajutorul căruia vom selecta unul din cele patru canale de ieșire, folosindu-ne de valoarea sa

Date de ieșire:

out_div – un vector cu 8 biți de tipul STD_LOGIC (7 downto 0) și reprezintă cele 7 segmente , cu ajutorul acestuia vom afisa cifrele corespunzatoare pe unul din cele patru canale de iesire

Semnale:

secunde - un vector de 6 biți de tipul std_logic_vector(5 downto 0) și reprezintă variabila corespunzătoare secundelor unui ceas în care se va contoriza trecerea unui interval de 1 secunde, luând valori în intervalul (0-60 respectiv 000000-011000 în binar)

minute - un vector de 6 biți de tipul std_logic_vector(5 downto 0) și reprezintă variabila corespunzătoare minutelor unui ceas în care se va contoriza trecerea unui interval de 60 secunde, luând valori în intervalul (0-60 respectiv 000000-011000 în binar)

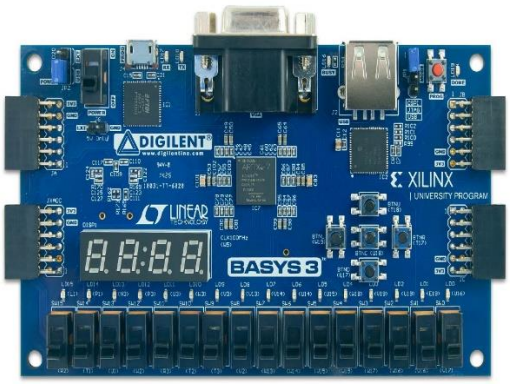
sec_counter – un vector de 24 biți de tipul unsigned(23 downto 0) utilizat pentru a interpreta corect intervalul de 1 secunde, ținând cont de faptul că placuța are o frecvență de 10 MHz; când se ajunge la valoare100000000se vor incrementa secundele

min_counter – un vector de 24 biți de tipul unsigned(23 downto 0) utilizat pentru a interpreta corect intervalul de 60 secunde, ținând cont de faptul că placuța are o frecvență de 10 MHz; când se ajunge la valoare 6000000000 se vor incrementa minutele

# 3.Metoda de implementare

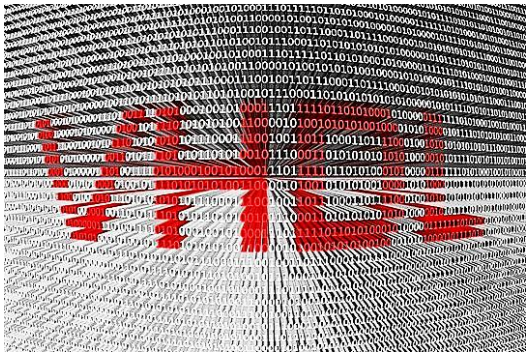
În cadrul acestui proiect au fost utilizate următoarele resurse:

1) circuitul FPGA xc7a35tcpg236-1, din familia ARTIX 7 fiind produs de XILINX, acest circuit fiind utilizat prin intermediul plăcii de dezvoltare Basys3 este utilizat pentru implementarea de soluții flexibile și scalabile în domeniul ingineriei electronice și poate fi programat pentru a realiza diferite funcții și algoritmi, oferind astfel un nivel ridicat de flexibilitate și adaptabilitate în proiectarea sistemelor electronice.




2) Vivado este un tool de sinteză și implementare FPGA dezvoltat de Xilinx, care facilitează proiectarea și dezvoltarea de circuite logice programabile. Acesta oferă o suită de instrumente puternice și eficiente pentru proiectarea, simularea, testarea și implementarea sistemelor digitale pe baza FPGA.


3) VHDL (Very High Speed Integrated Circuit Hardware Description Language) este un limbaj de descriere a hardware-ului utilizat în proiectarea și simularea circuitelor digitale. A fost dezvoltat pentru a permite inginerilor să specifice comportamentul și interconexiunea componentelor unui sistem electronic complex.



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

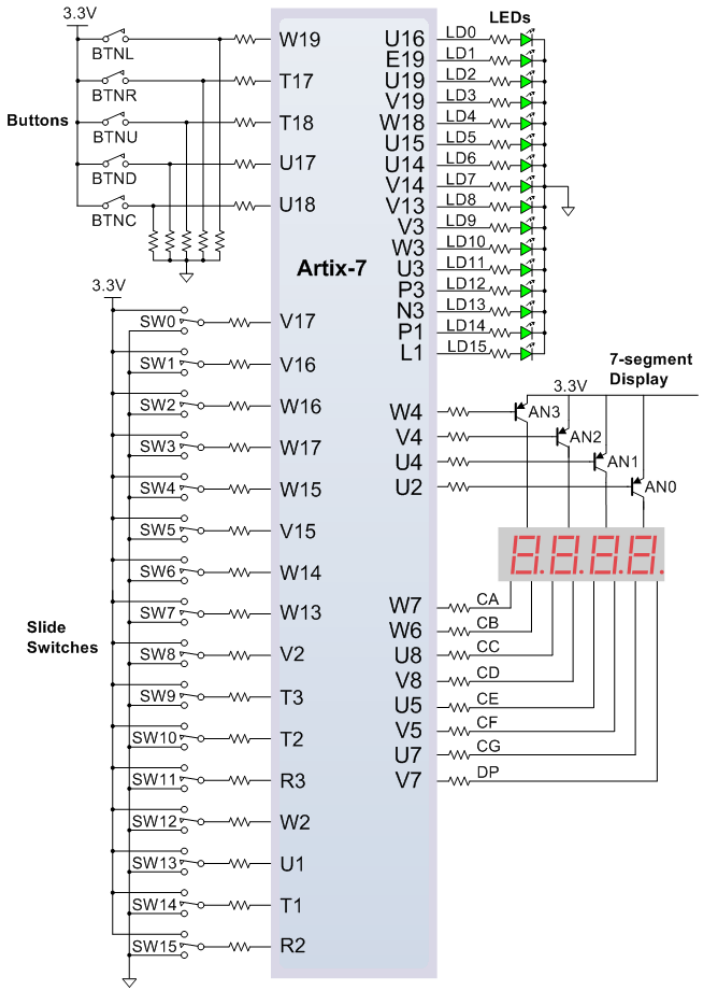
BASYS 3 este un sistem de dezvoltare hardware FPGA (Field-Programmable Gate Array) produs de Digilent. Este proiectat pentru a fi un instrument versatil și accesibil pentru dezvoltarea și testarea de circuite digitale.

BASYS 3 este bazat pe FPGA-ul de la Xilinx și vine echipat cu o varietate de caracteristici și componente. Printre acestea se numără:

- Un set bogat de porturi de intrare/ieșire, inclusiv porturi digitale, analogice și de comunicare, care permit conectarea și interacțiunea cu diverse dispozitive externe.

- Un ecran LCD și butoane pentru interacțiunea cu utilizatorul și afișarea informațiilor. - Conectori și interfețe de comunicare, cum ar fi USB, Ethernet, UART și SPI, pentru a permite comunicarea cu alte dispozitive sau calculatoare.

- Un sistem de alimentare integrat, care facilitează utilizarea și testarea circuitor în diverse scenarii de alimentare.



# 5. Editarea fișierului VHDL

Sapte Segmente – cod

```
----------------------------------------------

-- Module Name: ssd

----------------------------------------------


library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity ssd is

Port (

   digit0: IN STD_LOGIC_VECTOR(3 downto 0);

   digit1: IN STD_LOGIC_VECTOR(3 downto 0);

   digit2: IN STD_LOGIC_VECTOR(3 downto 0);

   digit3: IN STD_LOGIC_VECTOR(3 downto 0);

   clk: IN STD_LOGIC;

   out_div: OUT STD_LOGIC_VECTOR(6 downto 0);

   an: OUT STD_LOGIC_VECTOR(3 downto 0)

   );

end ssd;


architecture Behavioral of ssd is

signal cnt: STD_LOGIC_VECTOR(15 downto 0):=x"0000";
```

```
signal mux_digit: STD_LOGIC_VECTOR(3 downto 0);


begin


process(clk)
begin
   if(clk='1' and clk'event) then
      cnt<=cnt+1;
    end if;
end process;


process(cnt(15 downto 14), digit0, digit1, digit2, digit3)
begin
   case(cnt(15 downto 14)) is
      when "00" => mux_digit<=digit0;
      when "01" => mux_digit<=digit1;
      when "10" => mux_digit<=digit2;
      when "11" => mux_digit<=digit3;
      when others =>
    end case;
end process;
process(cnt(15 downto 14))
begin
   case(cnt(15 downto 14)) is
      when "00" => an<="1110";
```

```
        when "01" => an<="1101";
        when "10" => an<="1011";
        when "11" => an<="0111";
        when others =>
     end case;
end process;
with mux_digit select
   out_div<= "1111001" when "0001",   --1
         "0100100" when "0010",   --2
         "0110000" when "0011",   --3
         "0011001" when "0100",   --4
         "0010010" when "0101",   --5
         "0000010" when "0110",   --6
         "1111000" when "0111",   --7
         "0000000" when "1000",   --8
         "0010000" when "1001",   --9
         "0001000" when "1010",   --A
         "0000011" when "1011",   --b
         "1000110" when "1100",   --C
         "0100001" when "1101",   --d
         "0000110" when "1110",   --E
         "0001110" when "1111",   --F
         "1000000" when others;   --0
end Behavioral;
```

Decodor

```
----------------------------------------

-- Module Name: Decoder

----------------------------------------


library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;


entity decoder is

Port (

 sec: in std_logic_vector(5 downto 0);

 min: in std_logic_vector(5 downto 0);


digit0: out std_logic_vector(3 downto 0);

digit1: out std_logic_vector(3 downto 0);

digit2: out std_logic_vector(3 downto 0);

digit3: out std_logic_vector(3 downto 0));

end decoder;


architecture Behavioral of decoder is


begin
```

```
output_process: process(sec, min)
begin
   --reguli generale de obtinere a cifrelor
   digit0 <= conv_std_logic_vector((conv_integer(sec) mod 10), 4);
   digit1 <= conv_std_logic_vector((conv_integer(sec)/10) , 4);
   digit2 <= conv_std_logic_vector((conv_integer(min) mod 10), 4);
   digit3 <= conv_std_logic_vector((conv_integer(min)/10) , 4);
end process;

end Behavioral;
```

Contor ceas

```
----------------------------------------------
-- Module Name: Program ceas digital
----------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;

entity ceas_digital is
  Port (
```

```
        clk : in STD_LOGIC;
        reset : in STD_LOGIC;
        sec_out:out std_logic_vector(5 downto 0);
        min_out:out std_logic_vector(5 downto 0));

end ceas_digital;

architecture Behavioral of ceas_digital is

   signal secunde: std_logic_vector(5 downto 0) := "000000";
   signal minute: std_logic_vector(5 downto 0) := "000000";
   signal sec_counter : unsigned(31 downto 0) := (others => '0');
   signal min_counter : unsigned(31 downto 0) := (others => '0');

begin

process(clk, reset)
   begin
      if reset = '1' then
         secunde <= "000000";
         minute <= "000000";
         sec_counter <= (others => '0');
         min_counter <= (others => '0');

      elsif rising_edge(clk) then
```

```
        sec_counter <= sec_counter + 1;

        if sec_counter = 100000000 then
           secunde <= std_logic_vector(unsigned(secunde) + 1);
           sec_counter <= (others => '0');

           if secunde = "111001" then
              secunde <= "000000";
              min_counter <= min_counter + 1;
              minute <= std_logic_vector(unsigned(minute) + 1);
              if minute = "111001"then
                 minute <= "000000";
                 min_counter <= (others => '0');
              end if;

           end if;

        end if;

     end if;
  end process;

  sec_out<=secunde;
  min_out<=minute;
```

```
end Behavioral;
```

Programul principal

```
----------------------------------------------
-- Module Name: Top Module
----------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
library UNISIM;
use UNISIM.VComponents.all;
entity top_module is
Port (
        clk : in STD_LOGIC;
         reset : in STD_LOGIC;
         an: inout STD_LOGIC_VECTOR(3 downto 0);
         out_div : out STD_LOGIC_VECTOR(6 downto 0));
end top_module;
Architecture behavior of top_module Is
component ceas_digital is
  Port (
       clk : in STD_LOGIC;
         reset : in STD_LOGIC;
         sec_out:out std_logic_vector(5 downto 0);
```

```
        min_out:out std_logic_vector(5 downto 0));
end component;

component decoder is
Port (

 sec: in std_logic_vector(5 downto 0);
 min: in std_logic_vector(5 downto 0);

digit0: out std_logic_vector(3 downto 0);
digit1: out std_logic_vector(3 downto 0);
digit2: out std_logic_vector(3 downto 0);
digit3: out std_logic_vector(3 downto 0));
end component;

component ssd is
Port (
  digit0: IN STD_LOGIC_VECTOR(3 downto 0);
    digit1: IN STD_LOGIC_VECTOR(3 downto 0);
    digit2: IN STD_LOGIC_VECTOR(3 downto 0);
    digit3: IN STD_LOGIC_VECTOR(3 downto 0);
    clk: IN STD_LOGIC;
    out_div: OUT STD_LOGIC_VECTOR(6 downto 0);
    an: OUT STD_LOGIC_VECTOR(3 downto 0));
end component;
```

```
signal sec_out: std_logic_vector(5 downto 0):= (others => '0');
signal min_out: std_logic_vector(5 downto 0):= (others => '0');
signal digit0: std_logic_vector(3 downto 0);
signal digit1: std_logic_vector(3 downto 0);
signal digit2: std_logic_vector(3 downto 0);
signal digit3: std_logic_vector(3 downto 0);
begin
avg: ceas_digital port map(
clk=>clk,
reset=>reset,
sec_out=>sec_out,
min_out=>min_out

);

decoding: decoder port map(
 sec=>sec_out,
 min=> min_out,
 digit0 => digit0,
 digit1 => digit1,
 digit2 => digit2,
 digit3 => digit3
);
seven_segment_display: ssd port map(
```

```
 digit0 => digit0,
 digit1 => digit1,
 digit2 => digit2,
 digit3 => digit3,
 clk => clk,
 out_div => out_div,
 an => an
);
end behavior;
```

# 6. Editarea fișierului cu constrângeri

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
```

```
##7 segment display
set_property PACKAGE_PIN W7 [get_ports {out_div[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[0]}]
set_property PACKAGE_PIN W6 [get_ports {out_div[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[1]}]
set_property PACKAGE_PIN U8 [get_ports {out_div[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[2]}]
set_property PACKAGE_PIN V8 [get_ports {out_div[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[3]}]
set_property PACKAGE_PIN U5 [get_ports {out_div[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[4]}]
set_property PACKAGE_PIN V5 [get_ports {out_div[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[5]}]
set_property PACKAGE_PIN U7 [get_ports {out_div[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {out_div[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
    #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
```

```
set_property PACKAGE_PIN U18 [get_ports reset]
    set_property IOSTANDARD LVCMOS33 [get_ports reset]
```

# 7. Descrierea paşilor de sinteză şi testarea circuitului rezultat

1. Sinteză:

- Crearea unui proiect în Vivado și importarea fișierele VHDL ale circuitului.

- Adăugarea fișierul de constrângeri .xdc la proiect pentru a asocia semnalele circuitului cu pini specifici ai plăcii de dezvoltare.

2. Implementare:

- În implementare, Vivado va plasa și ruta componentele circuitului pe FPGA.

- Verificarea și optimizarea rutării pentru a asigura o propagare adecvată a semnalelor și o performanță optimă.

- Generarea fișierul bitstream, care conține configurația FPGA pentru circuitul sintetizat.

3. Programarea FPGA:

- Utilizarea programului Vivado pentru a încărca fișierul bitstream în placa FPGA.

4. Testarea în simulare:

- Crearea testbench-ului în Vivado sau altă unelte similare pentru a simula comportamentul circuitului.

- Definirea semnalele de intrare și verificarea rezultatele la semnalele de ieșire așteptate.

- Executarea simulării și analizarea rezultatelor pentru a verifica corectitudinea funcțională a circuitului.

5. Testarea pe placa FPGA:

- Încărcarea fișierului bitstream în placa FPGA și verificarea comportamentului circuitului în cadrul plăcii.
- Utilizarea instrumente pentru depanare și verificarea semnalelor pentru a valida funcționarea corectă a circuitului pe placa FPGA.

Acești pași asigură că circuitul este sintetizat, implementat și testat înainte de a fi utilizat în aplicații reale.

## 8.Concluzii

Conform celor prezentatea anterior, în cadrul proiectului la diciplina ELECTRONICA DIGITALA am crea un ceas care imită comportamentul unui ceas real.

Acesta numără, conform clock-ului prezent pe plăcuța FPGA intervale de timp de 1 secunde, respectiv intervale de timp de 60 secunde și le afișează continuu pe  Display-ul 7 segmente la intervalul de 1 secunde. Astfel, la final, vom obține pe plăcuța FPGA un ceas digital cu minute și secunde.

# Cuprins