

**Universitatea Tehnică “Gheorghe Asachi” din Iaşi**

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**ELECTRONICĂ DIGITALĂ**

**Proiect**

**Tema: MODUL TIMER0 – v1**

Studenţi:

Rusu Ioana-Simina

Lostun Șerban-Ilie

Leșan Vasile

Grupa : 1207B

Coordonator:

Asistent doctorand Ionica Pletea

**2023**

**Tema proiectului:**

**TIMER0 – v1**

**1. Specificaţiile proiectului:**

MODUL TIMER0 – v1

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) registrul TMR0 are dimensiunea de 8 biţi

b) sursa de clock clk\_in este cea a plăcii de dezvoltare

c) sursa de clock clk\_ex va fi generată de un buton extern

d) iniţializarea registrului TMR0 de la butoanele externe se face pentru EN=’1’

e) valoarea registrului TMR0 se va afisa în binar prin LED-uri

f) la depăşirea valorii maxime bitul „flag” TMRIF ia valoarea ‚1’



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

**2. Modulul TIMER0\_v1**

Modulul Timer0 este un *„timer”/„counter”* pe 8 biți având următoarele specificații:

● Registrul TMR0 (pe 8 biți)

● Un clock intern (cel de pe plăcuță) și un clock extern (implementat în cod)

● Un MUX 2x1 folosit pentru a selecta clock-ul dorit

● Un bit de flag folosit pentru a depista OVERFLOW-ul (depășirea capacității de reprezentare – în cazul nostru 8 biți)

Counter-ul nostru incrementează o valoare la fiecare front pozitiv al semnalului de clock. Alegerea clock-ului (cel intern – de pe plăcuță sau cel extern – dat de un buton) este făcută în cadrul procesului MUX 2x1, proces ce conține funcționalitatea unui multiplexor cu o selecție (*sel\_clk*) și 2 intrări (*clk\_div* și *clk\_ext*).

Ulterior este folosită variabila clk\_temp ce reprezintă clock-ul ales de noi.

INTRĂRILE în program vor fi:

● **DATA\_init** – un vector pe 8 biți generat de la butoanele externe ale plăcuței

● **EN** – un bit (FLAG ) folosit pentru a permite COUNTER-ului incrementarea valorii – activ pe *„1”* (front pozitiv)

● **btn\_reset** – un bit (FLAG) folosit pentru a reseta valoarea incrementată de COUNTER (această valoare va fi pusă pe 0) – activ pe *„1”*

● **sel\_clk** – un bit (FLAG) folosit drept selecție a clock-ului pe care dorim să-l utilizăm

● **clk\_in** – clock-ul de pe plăcuța de dezvoltare (ulterior se va folosi varianta divizată a acestuia, *clk\_div*)

● **btnClkEx** – clock-ul extern, generat de un buton de pe placă (butonul din centru )

IEȘIRILE din program vor fi:

● **TMR0** – valoarea incrementată de COUNTER, valoarea afișată în mod permanent la leduri

● **TMRIF** – un bit (FLAG) ce va lua valoarea *„1”* la depășirea valorii maxime (în momentul în care numărătoarea depășește valoarea „11111111” = FF = 255, se va aprinde un led atenționând acest lucru)

SEMNALELE programului vor fi:

● **TMR0\_v** – un vector pe 8 biți ce reprezintă registrul ce conține valoarea primită ca intrare

● **TMRIF\_flag** – 1 bit ce semnalizează depășirea capacității de reprezentare

● **clk\_div** – clock-ul divizat de pe placă (semnalul de ieșire din divizorul de frecvență)

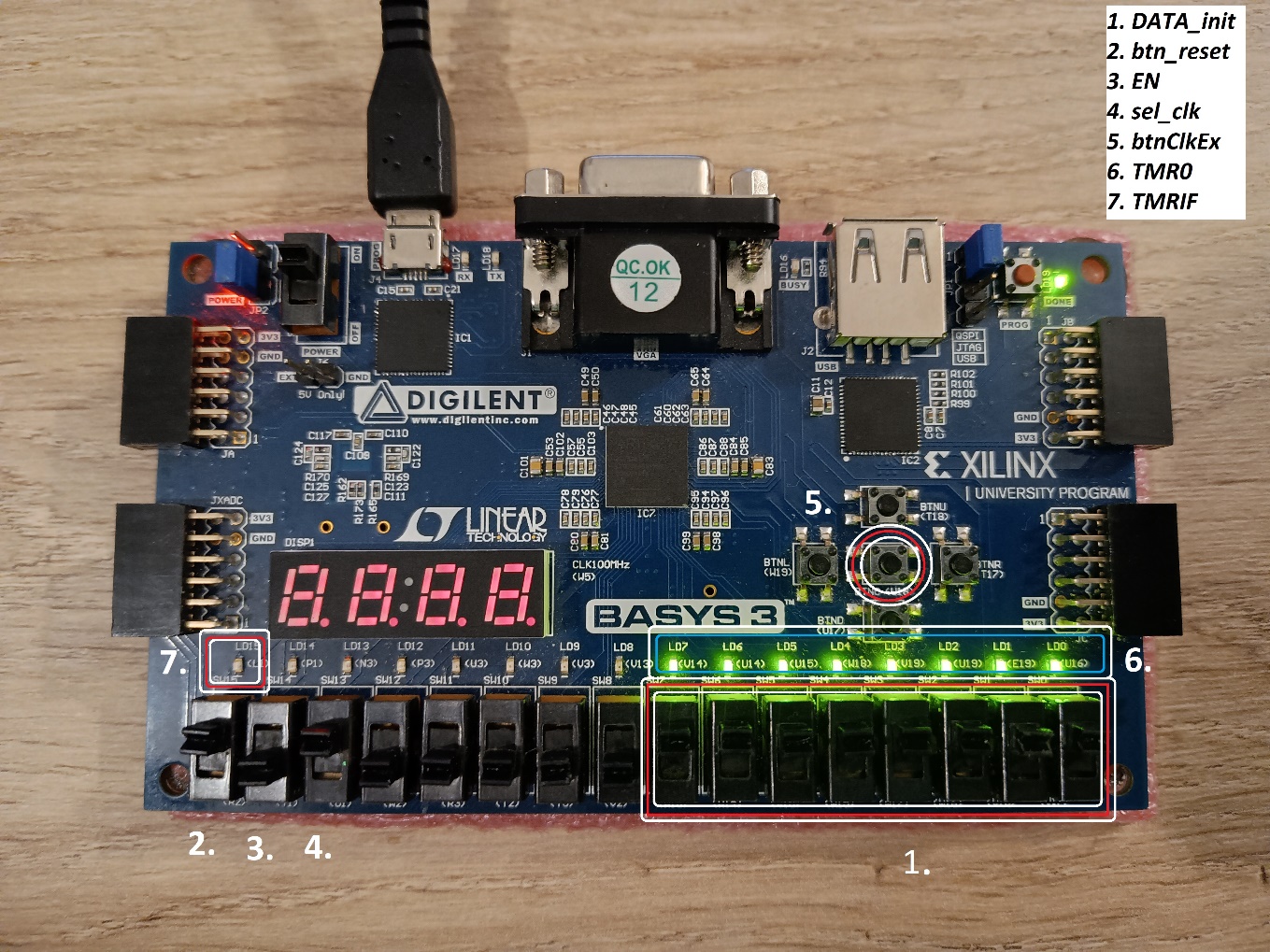
**3. Metoda de implementare**

În cadrul acestui proiect vor fi 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;
2. tool-ul de sinteză VIVADO;
3. limbajul de descriere VHDL (Very High Speed Integrated Circuit Hardware Description Language);

Implementarea proiectului a fost făcută printr-o descriere comportamentală. S-a proiectat entitatea **timer\_v1**. Aceasta are mai multe *„procese”*:

1. Un divizor de frecvență;
2. Un multiplexor 2x1;
3. Un numărător pe 8 biți;



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

A blue electronic board with a digital display

Description automatically generatedBASYS3 este o placă de dezvoltare FPGA (Field-Programmable Gate Array) produsă de Digilent. Placa BASYS3 conține un cip FPGA XILINX Artix-7, care oferă capacități de logică programabilă și resurse pentru implementarea diverselor proiecte digitale. Aceasta dispune de o gamă diversificată de resurse I/O, inclusiv butoane, LED-uri, conectori Pmod (pentru extensii modulare), conectori pentru afișaj cu șapte segmente, porturi USB, VGA și altele. Sistemul de dezvoltare este compatibil cu XILINX Vivado, un mediu integrat de dezvoltare (IDE) folosit pentru sinteza, implementarea și programarea FPGA-urilor și permite integrarea unui procesor MicroBlaze, care oferă o soluție de procesare soft-core. Aceste resurse facilitează interacțiunea cu mediul exterior și permite testarea proiectelor, fiind foarte accesibilă atât pentru studenți cât și pentru profesioniști.

Link-uri pentru vizualizarea funcționalității programului implementat:

* <https://youtu.be/Kc9aRXroP9M>
* <https://youtu.be/5UJveRpIU78>

**5. Editarea fişierului VHDL**

**library IEEE;**

**use IEEE.STD\_LOGIC\_1164.ALL;**

**use IEEE.std\_logic\_unsigned.ALL;**

**use IEEE.STD\_LOGIC\_ARITH.ALL;**

**entity timer\_v1 is**

**Port (**

**clk\_in : in std\_logic; -- clock ul placii**

**EN: in std\_logic; -- enable**

**btnClkEx: in std\_logic; -- clock extern**

**btn\_reset: in std\_logic; -- buton reset**

**sel\_clk: in std\_logic; -- clock selectie**

**TMRIF: out std\_logic; -- flag**

**TMR0 : out std\_logic\_vector (7 downto 0); -- registru pe 8 biti**

**DATA\_init: in STD\_LOGIC\_VECTOR(7 downto 0) -- vector mapat la switch-uri**

**);**

**end timer\_v1;**

**architecture Behavioral of timer\_v1 is**

**signal TMR0\_v: std\_logic\_vector(7 downto 0); -- registru pe 8 biti ce continue valoarea primita ca intrare**

**signal TMRIF\_flag: std\_logic := '0'; -- semnalul de flag**

**signal clk\_div: std\_logic; -- semnalul de clock de la iesirea divizorului de frecventa**

**signal clk\_temp: std\_logic; -- semnal de clock temporar**

**begin**

**-- Divizor de frecventa a clock-ului intern**

**process (clk\_in)**

**variable n: integer range 0 to 100000000;**

**begin**

**if clk\_in' event and clk\_in = '1' then**

**if n < 50000000 then**

**n := n+1;**

**else**

**n := 0;**

**end if;**

**if n < 25000000 then**

**clk\_div <= '1';**

**else**

**clk\_div <= '0';**

**end if;**

**end if;**

**end process;**

**-- MUX 2:1**

**process(EN, btn\_reset, btnClkEx, TMR0\_v, clk\_div, TMRIF\_flag, sel\_clk)**

**begin**

**case sel\_clk is**

**when '0' => clk\_temp <= clk\_div;**

**when '1' => clk\_temp <= btnClkEx;**

**end case;**

**end process;**

**-- Timer/Numarator**

**process(EN, btn\_reset, TMR0\_v, TMRIF\_flag)**

**begin**

**if EN = '1' then**

**if btn\_reset = '1' then**

**TMR0\_v <= DATA\_init;**

**TMRIF\_flag <= '0';**

**else**

**TMR0\_v <= DATA\_init;**

**if clk\_temp' event and clk\_temp = '1' then**

**if TMR0\_v = "11111111" then**

**TMRIF\_flag <= '1';**

**else**

**TMR0\_v <= TMR0\_v + 1;**

**end if;**

**end if;**

**end if;**

**end if;**

**if btn\_reset = '1' then**

**TMR0\_v <= DATA\_init;**

**TMRIF\_flag <= '0';**

**end if;**

**end process;**

**TMR0<= TMR0\_v;**

**TMRIF <= TMRIF\_flag;**

**end Behavioral;**

**6. Editarea fişierului de constrângeri**

**## Clock signal**

**set\_property PACKAGE\_PIN W5 [get\_ports clk\_in]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports clk\_in]**

**##create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports clk]**

**## Switches**

**set\_property PACKAGE\_PIN V17 [get\_ports {DATA\_init[0]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[0]}]**

**set\_property PACKAGE\_PIN V16 [get\_ports {DATA\_init[1]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[1]}]**

**set\_property PACKAGE\_PIN W16 [get\_ports {DATA\_init[2]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[2]}]**

**set\_property PACKAGE\_PIN W17 [get\_ports {DATA\_init[3]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[3]}]**

**set\_property PACKAGE\_PIN W15 [get\_ports {DATA\_init[4]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[4]}]**

**set\_property PACKAGE\_PIN V15 [get\_ports {DATA\_init[5]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[5]}]**

**set\_property PACKAGE\_PIN W14 [get\_ports {DATA\_init[6]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[6]}]**

**set\_property PACKAGE\_PIN W13 [get\_ports {DATA\_init[7]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {DATA\_init[7]}]**

**set\_property PACKAGE\_PIN U1 [get\_ports {sel\_clk}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {sel\_clk}]**

**set\_property PACKAGE\_PIN T1 [get\_ports {EN}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {EN}]**

**set\_property PACKAGE\_PIN R2 [get\_ports {btn\_reset}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {btn\_reset}]**

**## LEDs**

**set\_property PACKAGE\_PIN U16 [get\_ports {TMR0[0]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[0]}]**

**set\_property PACKAGE\_PIN E19 [get\_ports {TMR0[1]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[1]}]**

**set\_property PACKAGE\_PIN U19 [get\_ports {TMR0[2]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[2]}]**

**set\_property PACKAGE\_PIN V19 [get\_ports {TMR0[3]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[3]}]**

**set\_property PACKAGE\_PIN W18 [get\_ports {TMR0[4]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[4]}]**

**set\_property PACKAGE\_PIN U15 [get\_ports {TMR0[5]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[5]}]**

**set\_property PACKAGE\_PIN U14 [get\_ports {TMR0[6]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[6]}]**

**set\_property PACKAGE\_PIN V14 [get\_ports {TMR0[7]}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMR0[7]}]**

**set\_property PACKAGE\_PIN P1 [get\_ports {TMRIF}]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports {TMRIF}]**

**##Buttons**

**set\_property PACKAGE\_PIN U18 [get\_ports btnClkEx]**

**set\_property IOSTANDARD LVCMOS33 [get\_ports btnClkEx]**

**7. Descrierea paşilor de sinteză şi testarea circuitului rezultat**

Parcurgerea unui „flow” presupune următorii paşi:

1. S-a creat un proiect nou în programul „Vivado”;
2. S-a implementat modulul „timer\_v1” printr-o descriere comportamentală având drept „procese” un numărător, un divizor de frecvență și un multiplexor 2x1;
3. S-a editat fișierul de constrângeri ;
4. S-a sintetizat modulul (pentru a se vedea design-ul sintetizat);
5. S-a lansat implementarea proiectului care a avut ca efect final generarea fișierului bitstream;
6. S-a programat placa de dezvoltare BASYS3 cu fișierul bitstream și s-a testat funcționarea corespunzătoare a modulului implementat.

**8. Concluzii**

În concluzie, prin intermediul limbajului de programare VHDL (Very High-Speed Integrated Circuit Hardware Description Language) s-a implementat Modulul TIMER0\_v1. Funcționarea implementării proiectului a fost testată prin intermediul plăcii de dezvoltare BASYS3.

În urma testelor efectuate s-a constatat faptul că placa se comportă conform cerințelor.

**9. Bibliografie:**

1. PIC16LF1937, datasheet, <http://ww1.microchip.com/downloads/en/DeviceDoc/41364E.pdf> (pag. 191)

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

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