Universitatea Tehnică “Gheorghe Asachi” din Iaşi

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

ELECTRONICĂ DIGITALĂ

proiect

Temă: REG paralel-inel

Studenţi: Gălățanu Marco-Ionuț, Moloman Laurențiu-Ionuț, Petrișor Rareș-Gabriel

Grupa: 1207B

Coordonator:
Asist. Drd. Marius Obreja

2023

1. Specificaţiile proiectului:

Să se implementeze în FPGA prin descriere în limbaj VHDL, un sistem secvenţial: cu reset prioritar activ pe 0; cu două intrări de selecţie din care să se stabilească funcţionarea de registru paralel, respectiv, registru în inel cu deplasare stânga/ dreapta.

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

2. Funcţionalitate

Registrul are următoarele functionalități:

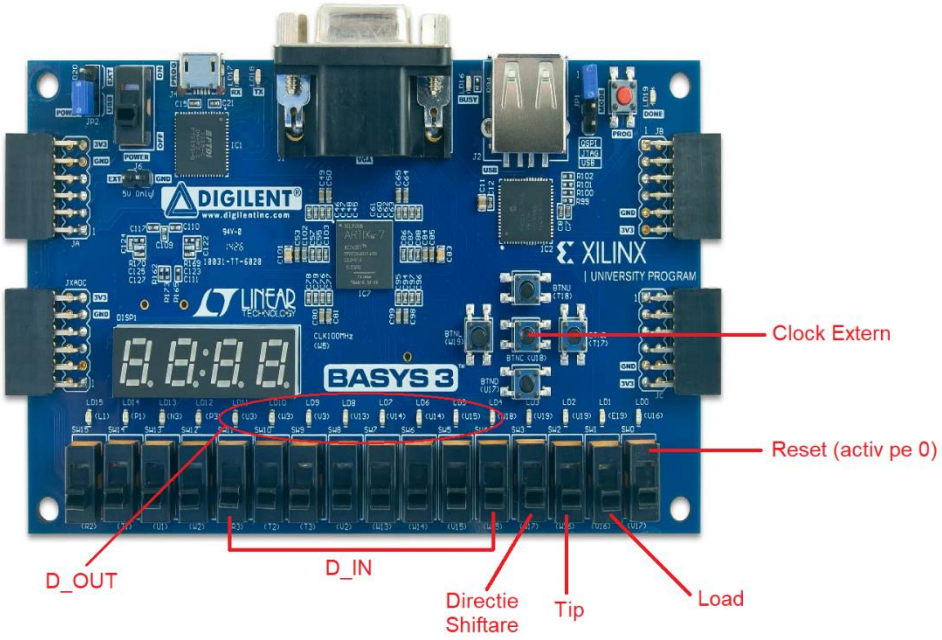
1. Registrul are ca dimensiune 8 biți
2. Memoria poate fi resetată (toți biții pe 0) de la switch-ul de reset V17
3. Switch-ul V16 (Load) permite incărcarea (pe 1) sau păstrarea informației (pe 0)
4. Putem alege tipul registrului: Registru Paralel (pe 0) sau Registru Inel (pe 1). Controlul se face prin intermediul switch-ului W16.
5. Switch-ul W17 ne permite să alegem direcția selectată pentru registrul inel (daca este 0, shiftarea se face de la stânga spre dreapta, invers in caz contrar).
6. Butonul U17 reprezintă clock-ul extern, functioneaza pe frontul pozitiv al clock-ului.
7. Switch-urile W15, V15, W14, W13, V2, T3, T2, R3 sunt pentru încărcarea datelor.
8. Led-urile W18, U15, U14, V14, V13, V3, W3, U3 sunt pentru afișarea datelor.

Mod de utilizare pentru registrul inel:

1. Trebuie să încărcăm prima dată datele selectând registrul paralel (pe 0), datele se vor reține în memorie in vectorul D. Apoi dăm switch pe registru inel (pe 1).
2. Selectăm direcția de deplasare/ shiftare.
3. Apasăm pe butonul de clock, iar biții se vor shifta/ deplasa circular in direcția selectată.
4. Informația reținută de registru este afișată pe LED-uri.
5. Prin intermediul reset-ului se pot reseta totii biții pe 0.

Mod de utilizare pentru registrul paralel:

1. Încărcăm prima dată datele selectând registrul paralel (pe 0), datele se vor reține în memorie in vectorul D.
2. Informația reținută de registru este afișată pe LED-uri.



3. Metoda de implementare

Pentru implementarea acestui modul s-au folosit programul de sinteză Vivado și limbajul VHDL. Implementarea proiectului a fost facută printr-o descriere comportamentală. S-a proiectat entitatea. 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 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 colectie 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 controller-ele embedded.

5. Editarea fişierului VHDL

Entitatea și arhitectura registrului:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VComponents.all;

--Register entity
entity pipo_inel_register is
    Port ( clk : in  STD_LOGIC;
           swt : in  STD_LOGIC; -- bit pentru directie incarcare
           load : in std_logic; -- bit pentru incarcare date
           tip : in std_logic; -- bit de selectie siso sau pipo
           D_IN : in std_logic_vector(7 downto 0);
           D_OUT : out std_logic_vector(7 downto 0);
           reset : in std_LOGIC
            );
end pipo_inel_register;

--Register architecture
architecture Behavioral of pipo_inel_register is
    signal D: std_logic_vector(7 downto 0);
    signal Q: std_logic_vector(7 downto 0);
begin
    D <= D_IN;
    D_OUT <= Q;
    -- update internal register on rising edge of clock
    process (clk)
        variable aux : std_logic;
    begin
```

Funcționarea registrului in VHDL:

```vhdl
begin
    D <= D_IN;
    D_OUT <= Q;
    -- update internal register on rising edge of clock
    process (clk)
        variable aux : std_logic;
    begin
      if (reset = '1') then
        if rising_edge(clk) then
           if (load = '1') then
                if( tip = '0') then
                    Q <= D;

                else
                    if (swt = '0') then --incarcare stanga->dreapta--Q
                       aux:=  Q(7);
                       Q(7) <= Q(6) ;
                       Q(6) <= Q(5) ;
                       Q(5) <= Q(4) ;
                       Q(4) <= Q(3) ;
                       Q(3) <= Q(2) ;
                       Q(2) <= Q(1) ;
                       Q(1) <= Q(0) ;
                       Q(0) <= aux ;

                    else -- incarcare dreapta->stanga
                       aux:= Q(0);
                       Q(0) <= Q(1) ;
                       Q(1) <= Q(2) ;
                       Q(2) <= Q(3) ;
                       Q(3) <= Q(4) ;
                       Q(4) <= Q(5) ;
                       Q(5) <= Q(6) ;
                       Q(6) <= Q(7) ;
                       Q(7) <= aux ;

                    end if;
                end if;
           end if;
        end if;
      else
            Q<= (others=>'0');
      end if;

    end process;

end Behavioral;
```

6. Editarea fişierului de constrângeri

Clock-ul extern al plăcii:

```
set_property PACKAGE_PIN U18 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk]
```

Switch-uri:

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {reset}]
    set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
set_property PACKAGE_PIN V16 [get_ports {load}]
    set_property IOSTANDARD LVCMOS33 [get_ports {load}]
set_property PACKAGE_PIN W16 [get_ports {tip}]
    set_property IOSTANDARD LVCMOS33 [get_ports {tip}]
set_property PACKAGE_PIN W17 [get_ports {swt}]
    set_property IOSTANDARD LVCMOS33 [get_ports {swt}]
set_property PACKAGE_PIN W15 [get_ports {D_IN[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[0]}]
set_property PACKAGE_PIN V15 [get_ports {D_IN[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[1]}]
set_property PACKAGE_PIN W14 [get_ports {D_IN[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[2]}]
set_property PACKAGE_PIN W13 [get_ports {D_IN[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[3]}]
set_property PACKAGE_PIN V2 [get_ports {D_IN[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[4]}]
set_property PACKAGE_PIN T3 [get_ports {D_IN[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[5]}]
set_property PACKAGE_PIN T2 [get_ports {D_IN[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[6]}]
set_property PACKAGE_PIN R3 [get_ports {D_IN[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_IN[7]}]
```

LED-uri:

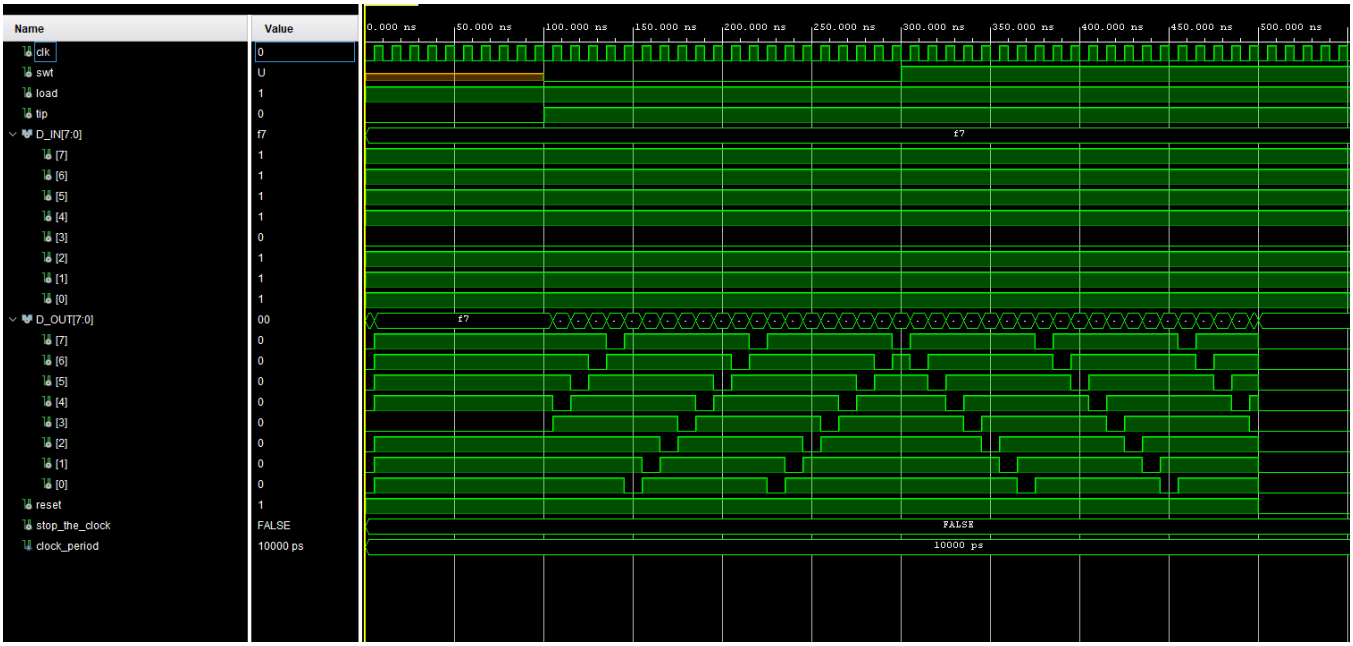
```
## LEDs
#set_property PACKAGE_PIN V19 [get_ports {D_OUT[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[0]}]
set_property PACKAGE_PIN W18 [get_ports {D_OUT[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[0]}]
set_property PACKAGE_PIN U15 [get_ports {D_OUT[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[1]}]
set_property PACKAGE_PIN U14 [get_ports {D_OUT[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[2]}]
set_property PACKAGE_PIN V14 [get_ports {D_OUT[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[3]}]
set_property PACKAGE_PIN V13 [get_ports {D_OUT[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[4]}]
set_property PACKAGE_PIN V3 [get_ports {D_OUT[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[5]}]
set_property PACKAGE_PIN W3 [get_ports {D_OUT[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[6]}]
set_property PACKAGE_PIN U3 [get_ports {D_OUT[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D_OUT[7]}]
#set property PACKAGE PIN V3 [get ports {led[9]}]
```

7. Descrierea paşilor de sinteză şi testarea circuitului rezultat

1. S-a creat un proiect nou în programul Vivado.
2. S-a implementat entitatea registru “pipo_inel_register“ printr-o descriere comportamentală, având drept “procese” un registru cu 2 funcții: registru paralel sau registru inel.
3. S-a editat fișierul de constrângeri în vederea realizării legăturilor între switch-uri și intrări (reset, load, tip, swt, D_IN(7:0)), butonul din mijloc reprezintă clock-ul extern, ieșirea D_OUT(7:0) cu LED-urile care corespund pozițiilor switch-urilor D_IN.
4. S-a realizat analiza RTL (Register Transfer Level).
5. S-a sintetizat modulul, pentru a 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 modulului:

Registru PIPO-Inel (cu toate funcționalitățile sale).



9. Concluzii

În concluzie s-a implementat în FPGA prin descriere în limbaj VHDL, un sistem secvenţial cu reset prioritar activ pe 0, cu încărcare cu două intrări de selecţie din care s-a stabilit funcționarea de registru paralel (tip), respectiv, registru în inel cu deplasare stânga/ dreapta (swt).

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-manual