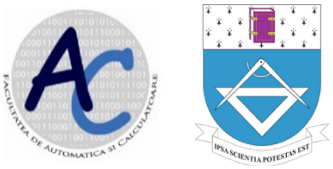


Universitatea Tehnică "Gheorghe Asachi" din Iaşi

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

ELECTRONICĂ DIGITALĂ

proiect

Tema: REG paralel - inel - v1

2020

1. Specificaţiile proiectului:

REG paralel - inel - v1

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 funcţionare 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.

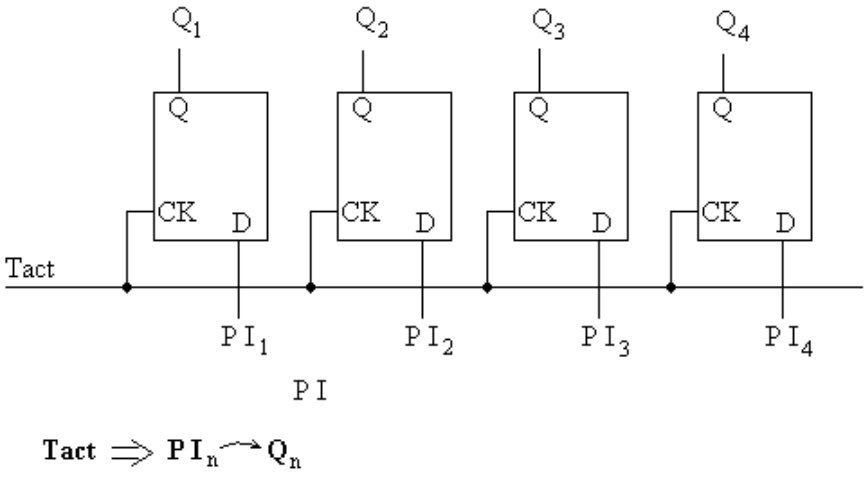
2. Modulul

- se va descrie în câteva fraze funcţionalităţile modulului

Regiştrii sunt circute digitale secvenţiale de stocare a informaţiei sub formă binară ce permit scrierea sau citirea simultană a tuturor biţilor.

Din punct de vedere structural, regiștri, sunt constituiţi din mai multe circute de tip flip-flop ce sunt grupate împreună. Descrierea unui proces secvenţial in limbaj VHDL se face prin urmărirea evenimentelor unui semnal de tact (clk).

Un registru paralel constă de fapt într-un număr de bistabile de tip D care au **tactul în comun.**

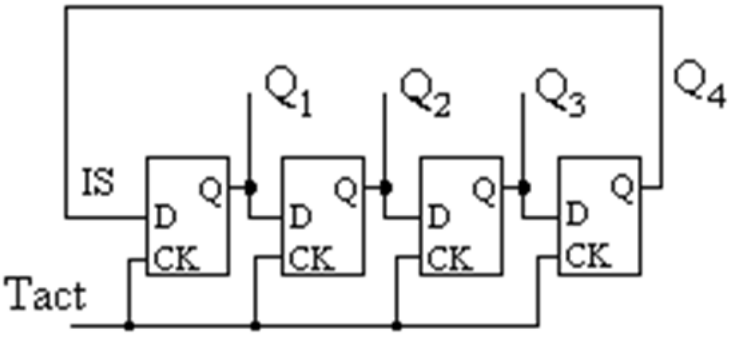


La aplicarea tactului fiecare bistabil va încărca data de la propria intrare (fiecare circuit va acţiona conform funcţionării unui bistabil tip D). Se remarcă sincronismul acţiunii de încărcare de unde şi numele ansamblului care este registru.

Intrările D ale bistabilelor constituie intrarea registrului iar ieşirile bistabilelor sunt ieşirea registrului. Intrările bistabilelor poartă numele de intrări paralele, „parallel input”, de unde şi notarea PI. Evident, la aplicarea tactului, data de la intrarea PI cu indicele n va fi transferată la ieşirea Q cu acelaşi indice, de unde denumirea de „încărcare paralel”(parallel load).

Registrul in inel reprezinta o aplicatie a registrelor in serie. Structura circuitului este aceeaşi cu cea prezentată la registrul paralel (un şir de bistabile D care au **tactul comun**) cu deosebirea că fiecare bistabil **are intrarea D conectată la ieşirea Q a precedentului.**

Intrarea D a primului bistabil poartă numele de **intrare serie**, notată IS (evident este singura intrare D a vreunui bistabil care este accesibilă utilizatorului). Ieşirile Q ale bistabilelor constituie ieşirea registrului.



Registru in inel

3. Metoda de implementare

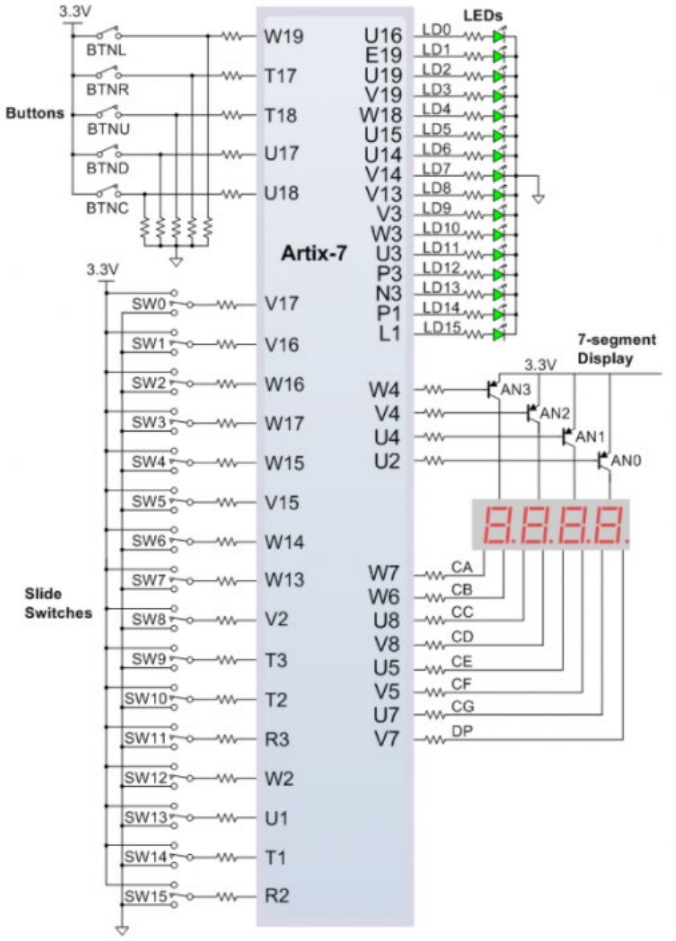
Utilizarea resurselor: circuit FPGA, limbajul VHDL, programul de sinteză Vivado

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

Basys 3 este o platforma de dezvoltare a circuitelor digitale bazata Artix-7 FPGA (field programmable gate array) de la Xilinx.

Basys3 oferă o colecție de porturi și periferice precum: 16 comutatoare utilizator (switchuri), 16 LED-uri de utilizator, 5 butoane utilizator ,Afișaj din 4 cifre cu 7 segmente, USB HID Host pentru mouse, tastaturi și stick-uri de memorie si multe altele.

Artix-7 ofera urmatoarele functii: 33.280 celule logice in 5200 unitati, 1800 Kbits de memorie rapida RAM, 5 clock-uri de gestionare cu phase-locked loop (PLL), 90 unitati DSP, Viteze ale clock-ului intern de peste 450MHz, Convertor de la analog la digital.



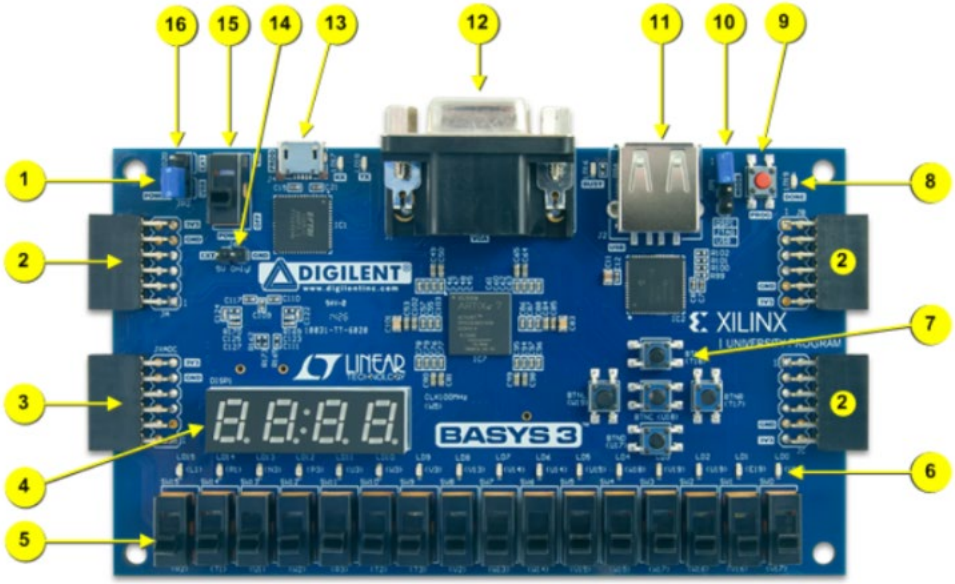


Figure 1. Basys3 board features

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

```
1  library ieee;
2
3  library work;
4
5  use ieee.std_logic_1164.all;
6
7  use ieee.std_logic_misc.all;
8
9  use ieee.numeric_std.all;
10
11 use ieee.std_logic_unsigned.all;
12
13 entity reg_par is
14
15      port (
16
17         selectie : in std_logic;
18         selectie_deplasare: in std_logic;
19
20         reset :  in std_logic;
21
22         en :  in std_logic;
23
24         clk :  in std_logic;
```

```
        data_init :  in std_logic_vector( 7  downto 0  );
        result :  out std_logic_vector( 7  downto 0  ) ;
        bitin : in STD_LOGIC
    );
end entity;



architecture rtl of reg_par is
    signal aux: std_logic_vector(7 downto 0);
    signal clk_in : std_logic;
    signal Data : STD_LOGIC_VECTOR(7 downto 0) := "00000000";

    begin
     process(clk)
        variable n : integer range 0 to 1000000000;
        begin
        if clk'event and clk='1' then
            if n < 100000000 then
                n := n+1;
            else
                 n := 0;
            end if;
            if n <= 50000000 then
                clk_in <= '1';
             else
                 clk_in <= '0';
             end if;
         end if;
    end process;
        process(clk_in,reset,en,data_init, selectie)
        begin
```

```
87        if selectie='0' then
88
89            if reset = '0' then
90
91                aux<= "00000000" ;
92
93                else
94
95                    if (clk_in'EVENT and clk_in = '1') then
96
97                            aux <= data_init;
98
99
100                   end if;
101
102               end if;
103
104               result<=aux;
105
106         elsif selectie='1' then
107
108           if(reset = '0') then    -- reset
109               Data <= "00000000";
110           else
111               if(rising_edge(clk_in))then
112                   if(selectie_deplasare = '0') then --deplasare stanga
113
114                       Data(7) <= Data(6);
115                       Data(6) <= Data(5);
116                       Data(5) <= Data(4);
117                       Data(4) <= Data(3);
118
119                       Data(3) <= Data(2);
120                       Data(2) <= Data(1);
121                       Data(1) <= Data(0);
122                       Data(0) <= bitin;
123                   else    --deplasare dreapta
124                       Data(0) <= Data(1);
125                       Data(1) <= Data(2);
126                       Data(2) <= Data(3);
127                       Data(3) <= Data(4);
128
129                       Data(4) <= Data(5);
130                       Data(5) <= Data(6);
131                       Data(6) <= Data(7);
132                       Data(7) <= bitin;
133
134                   end if;
135               end if;
136           end if;
137         result<=Data;
138         end if;
139
140
141
142       end process;
143
144   end rtl;
```

6. Editarea fişierului de 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]
 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets 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 W2 [get_ports {bitin}]

set_property IOSTANDARD LVCMOS33 [get_ports {bitin}]

set_property PACKAGE_PIN T1 [get_ports {selectie}]

set_property IOSTANDARD LVCMOS33 [get_ports {selectie}]
```

```
51
52  set_property PACKAGE_PIN U1 [get_ports {selectie_deplasare}]
53
54  set_property IOSTANDARD LVCMOS33 [get_ports {selectie_deplasare}]
55  set_property PACKAGE_PIN R2 [get_ports {reset}]
56
57  set_property IOSTANDARD LVCMOS33 [get_ports {reset}]
58
59  ## LEDs
60  set_property PACKAGE_PIN U16 [get_ports {result[0]}]
61
62  set_property IOSTANDARD LVCMOS33 [get_ports {result[0]}]
63
64  set_property PACKAGE_PIN E19 [get_ports {result[1]}]
65
66  set_property IOSTANDARD LVCMOS33 [get_ports {result[1]}]
67
68  set_property PACKAGE_PIN U19 [get_ports {result[2]}]
69
70  set_property IOSTANDARD LVCMOS33 [get_ports {result[2]}]
71
72  set_property PACKAGE_PIN V19 [get_ports {result[3]}]
73
74  set_property IOSTANDARD LVCMOS33 [get_ports {result[3]}]
75
76  set_property PACKAGE_PIN W18 [get_ports {result[4]}]
77
78  set_property IOSTANDARD LVCMOS33 [get_ports {result[4]}]
79
80  set_property PACKAGE_PIN U15 [get_ports {result[5]}]
81
82  set_property IOSTANDARD LVCMOS33 [get_ports {result[5]}]
83
84  set_property PACKAGE_PIN U14 [get_ports {result[6]}]
85
86  set_property IOSTANDARD LVCMOS33 [get_ports {result[6]}]
87
88  set_property PACKAGE_PIN V14 [get_ports {result[7]}]
89
90  set_property IOSTANDARD LVCMOS33 [get_ports {result[7]}]
```

7. Descrierea paşilor de sinteză şi testarea circuitului rezultat

Interfata modulului:

- reset: resetează, când are valoarea 0, valoarea întregului modul.

- selectie: selectează pentru valoarea 0 funcționalitatea de registru cu încărcare paralelă și pentru 1 pe cea cu încărcare în inel cu deplasare stânga/dreapta.

- selectie_deplasare: selectează pentru valoarea 0 deplasarea spre stânga și pentru 1 deplasarea spre dreapta a registrului în inel.

- **clk:** semnalul de clock al plăcuței.

- *data_init*: numărul binar pe 8 biți cu care se încarcă registrul paralel.

- **bitin:** pentru valoarea 1 pune in funcționare led-urile pentru registrul în inel cu deplasare stânga/dreapta.

- result: expune prin led-uri rezultatul încărcării paralele sau prin inel cu deplasare stânga/dreapta.

8. Concluzii

Regiștrii paralel au utilitatea de a păstra într-un circuit electronic o valoarea pentru un anumit timp, în timp ce regiștrii serie au utilitatea de a păstra într-un circuit mai multe valoari pentru un anumit timp. În aplicația prezentată am arătat funcționalitatea registrului paralel și cea a registrului în inel cu deplasare stânga/dreapta care realizează înscrierea și citirea bit cu bit, în timp ce deplasarea o realizează într-un singur sens, în funcție de selecția aleasă.

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