**11 July 2006**

Create a VHDL entity synchronous to a clock signal with 2 4-bits inputs:

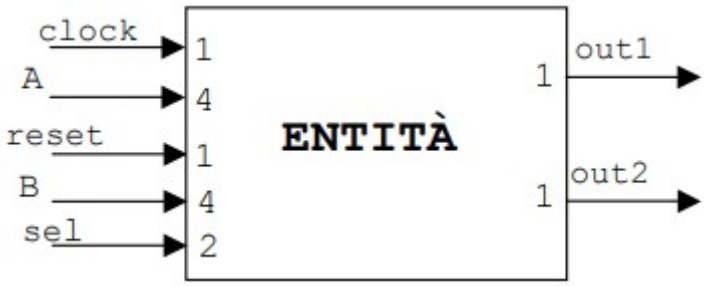
**A** (a3, a2, a1, a0)
**B** (b3, b2, b1, b0)

and 2 serial outputs: **out1** and **out2**

The entity must produce the output as described in the table below, depending on the value of a selection signal (**sel**):

| sel | out1 | out2 |
|---|---|---|
| 0 | High z | High z |
| 1 | a3 a2 a1 a0 | b3 b2 b1 b0 |
| 2 | b0 b1 b2 b3 | a0 a1 a2 a3 |
| 3 | b3 a0 b2 a1 b1 a2 b0 a3 | a3 b0 a2 b1 a1 b2 a0 b3 |

Avoid the accumulation of data inside the entity.
The entity must provide an asynchronous reset signal (**reset**) which reset the state of the entity and at the same time set the output to high impedance.



```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6
7   entity PF4 is
8      port(CLK, RST: in std_logic;
9           sel: in std_logic_vector (1 downto 0);
10          A, B: in std_logic_vector (3 downto 0);
11          out1, out2: out std_logic);
12  end PF4;
13
14
15  architecture Behavioral of PF4 is
16
17     signal cnt: integer range 0 to 15:=0;
18     signal reginA: std_logic_vector (3 downto 0):=(others => 'Z');
19     signal reginB: std_logic_vector (3 downto 0):=(others => 'Z');
20     signal regin1: std_logic_vector (7 downto 0):=(others => 'Z');
21     signal regin2: std_logic_vector (7 downto 0):=(others => 'Z');
22     signal regsel: std_logic_vector (1 downto 0):="ZZ";
23
24     begin
25
26        process (CLK, RST)
27        begin
28
29           if (RST='1') then
30              cnt <= 0;
31              reginA <= (others => 'Z');
32              reginB <= (others => 'Z');
33              regsel <= "ZZ";
34              out1 <= 'Z';
35              out2 <= 'Z';
36
37           elsif rising_edge(CLK) then
38              if (cnt=0) then
39                 reginA <= A;
40                 reginB <= B;
41                 regsel <= sel;
42
43                 regin1(7) <= B(3);
44                 regin1(6) <= A(0);
45                 regin1(5) <= B(2);
46                 regin1(4) <= A(1);
47                 regin1(3) <= B(1);
48                 regin1(2) <= A(2);
49                 regin1(1) <= B(0);
50                 regin1(0) <= A(3);
51
52                 regin2(7) <= A(3);
53                 regin2(6) <= B(0);
54                 regin2(5) <= A(2);
55                 regin2(4) <= B(1);
56                 regin2(3) <= A(1);
57                 regin2(2) <= B(2);
58                 regin2(1) <= A(0);
59                 regin2(0) <= B(3);
60
61                 out1 <= 'Z';
62                 out2 <= 'Z';
63                 cnt <= cnt + 1;
64
65
66
67
68
69
70
71
72
73
74
75
76
```

```
77              elsif (cnt > 0) then
78                 case regsel is
79                    when "00" => out1 <= 'Z';
80                                 out2 <= 'Z';
81
82                    when "01" => if (cnt <5) then
83                                    out1 <= reginA(3);
84                                    out2 <= reginB(3);
85                                    reginA <= reginA(2 downto 0) & 'Z';
86                                    reginB <= reginB(2 downto 0) & 'Z';
87                                    cnt <= cnt + 1;
88                                 elsif (cnt >= 5) then
89                                    out1 <= 'Z';
90                                    out2 <= 'Z';
91                                 end if;
92
93                    when "10" => if (cnt <5) then
94                                    out1 <= reginB(0);
95                                    out2 <= reginA(0);
96                                    reginA <= 'Z' & reginA(3 downto 1);
97                                    reginB <= 'Z' & reginB(3 downto 1);
98                                    cnt <= cnt + 1;
99                                 elsif (cnt >= 5) then
100                                   out1 <= 'Z';
101                                   out2 <= 'Z';
102                                end if;
103
104                   when "11" => if (cnt <9) then
105                                   out1 <= regin1(7);
106                                   out2 <= regin2 (7);
107                                   regin1 <= regin1(6 downto 0) & 'Z';
108                                   regin2 <= regin2(6 downto 0) & 'Z';
109                                   cnt <= cnt + 1;
110                                elsif (cnt >= 8) then
111                                   out1 <= 'Z';
112                                   out2 <= 'Z';
113                                end if;
114
115                   when others => out1 <= 'Z';
116                                  out2 <= 'Z';
117
118                end case;
119
120             end if;
121
122          end if;
123
124      end process;
125
126   end Behavioral;
```

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   --data input: 2 parallel 4 bits; 2 parallel bits (A and B);
7   --selection input: 2 parallel bits (SEL);
8   --output: 2 serial signals, out1 and out2 (bitstream process is explained later).
9
10  entity P_to_S is
11     port(CLK, RST: in std_logic;
12          SEL: in std_logic_vector(1 downto 0):="00";
13          A, B: in std_logic_vector (3 downto 0):="ZZZZ";
14          out1, out2: out std_logic);
15  end P_to_S;
16
17  architecture Behavioral of P_to_S is
18     signal cnt: integer range 0 to 10:=0;
19     signal regA, regB: std_logic_vector (3 downto 0):="ZZZZ";
20     signal regSEL: std_logic_vector (1 downto 0):="ZZ";
21  begin
22
23  process (CLK, RST)
24  begin
25  if (RST = '1') then
26     cnt <= 0;
27     regA <= "ZZZZ";
28     regB <= "ZZZZ";
29     out1 <= 'Z';
30     out2 <= 'Z';
31  elsif (rising_edge(CLK)) then
32     --1) upload A and B into regA and regB (cnt=0) and out1,2='Z'
33     --at the same time SEL input can be controlled; SEL might change
34     --while output bitstream is processed, thus the content of SEL
35     --is stored into regSEL just at this point;
36     if (cnt = 0) then
37        regSEL <= SEL;
38        regA <= A;
39        regB <= B;
40        out1 <= 'Z';
41        out2 <= 'Z';
42        cnt <= cnt + 1;
43     --2) at this point, regSEL is ready and its content can be controlled
44     --NB: if SEL=00, out1 and out2 are kept in high impedance state indefinitely;
45     --otherwise, after the output bitstreams end, the entity makes a new selection
46     --in other words, a new occurence of SEL is stored into regSEL.
47     elsif (cnt > 0) then
48        case regSEL is
49           --regSEL=00: output in high impedance state until a reset occurs;
50           when "00" => out1 <= 'Z';
51                        out2 <= 'Z';
52           --regSEL=01: out1=a3...10; out2=b3...b0.
53           when "01" => out1 <= regA(3);
54                        regA <= regA(2 downto 0) & 'Z';
55                        out2 <= regB(3);
56                        regB <= regB(2 downto 0) & 'Z';
57                        if (cnt = 4) then
58                          cnt <= 0;
59                        else
60                          cnt <= cnt + 1;
61                        end if;
62           --regSEL=10: out1=b0...b3; out2=a0...a3
63           when "10" => out1 <= regB(0);
64                        regB <= 'Z' & regB(3 downto 1);
65                        out2 <= regA(0);
66                        regA <= 'Z' & regA(3 downto 1);
67                        if (cnt = 4) then
68                          cnt <= 0;
69                        else
70                          cnt <= cnt + 1;
71                        end if;
72
73
74
75
76
```

```
77              --regSEL=11: out1: b3a0b2a1b1a2b0a3; out2: a3b0a2b1a1b2a0b3
78              when "11" => if (cnt = 1) then
79                             out1 <= regB(3);
80                             out2 <= regA(3);
81                             cnt <= cnt + 1;
82                           elsif (cnt = 2) then
83                             out1 <= regA(0);
84                             out2 <= regB(0);
85                             cnt <= cnt + 1;
86                           elsif (cnt = 3) then
87                             out1 <= regB(2);
88                             out2 <= regA(2);
89                             cnt <= cnt + 1;
90                           elsif (cnt = 4) then
91                             out1 <= regA(1);
92                             out2 <= regB(1);
93                             cnt <= cnt + 1;
94                           elsif (cnt = 5) then
95                             out1 <= regB(1);
96                             out2 <= regA(1);
97                             cnt <= cnt + 1;
98                           elsif (cnt = 6) then
99                             out1 <= regA(2);
100                            out2 <= regB(2);
101                            cnt <= cnt + 1;
102                          elsif (cnt = 7) then
103                            out1 <= regB(0);
104                            out2 <= regA(0);
105                            cnt <= cnt + 1;
106                          elsif (cnt = 8) then
107                            out1 <= regA(3);
108                            out2 <= regB(3);
109                            cnt <= 0;
110                          end if;
111             when others => cnt <= 0; --an error occurred in this case
112                                      --hopefully this won't happen.
113         end case;
114      end if;
115   end if;
116
117   end process;
118
119   end Behavioral;
120
```