30 January 2015

Create a clock-synchronous VHDL entity which starting from a reset samples the first 4 bits of 2 serial inputs **`DIN1`** and **`DIN2`**.

The entity must produce 4 outputs **`DOUT1`**, **`DOUT2`**, **`DOUT3`**, **`DOUT4`**, correspondent, respectively to the first, second, third, and fourth bits of the inputs with the following behaviour:

- High impedance during the sampling of the inputs.
- 0 for 2 clock cycles if the correspondent bits at the inputs are equal.
- 1 for 2 clock cycles if the correspondent bits at the inputs are not equal.
- They assume high impedance state again for 2 clock cycles.

After 4 clock cycles the analysis starts again

```
1   library IEEE; use IEEE.STD_LOGIC_1164.ALL;
2   use IEEE.STD_LOGIC_ARITH.ALL; use IEEE.STD_LOGIC_UNSIGNED.ALL;
3
4   entity PF is
5      port(CLK, DIN1, DIN2, RESET: in std_logic;
6         DOUT1, DOUT2, DOUT3, DOUT4: out std_logic);
7   end PF;
8
9   architecture Behavioral of PF is
10     signal regOut: std_logic_vector(3 downto 0):="ZZZZ";
11     signal EN: bit:='0';
12     signal cnt: integer range 0 to 10:=0;
13  begin
14  process(RESET, CLK)
15  begin
16     --1)RESET:
17     if (RESET = '1') then
18        regOut <= "ZZZZ"; EN <= '0';  cnt <= 0;
19     --RESET = '0':
20     elsif (rising_edge(CLK)) then
21        --2) For 4 clock periods of time serial input are compared:
22        --depending on the result of this operation, regOut is updated:
23        --regOut(0) contains the result of comparation for the first (DIN1, DIN2) pair;
24        --regOut (1) contains the result of comparation for the second (DIN1, DIN2) pair;
25        --and so on...
26        --At the same time, a signal "EN" is kept at low level so that it can prevent regOut content to be
27        --tranferred to the output signals.
28        --when cnt = 3 (last clock period of 4), EN is set to '1' so that during the following 2 clock periods
29        --of time output signals can be tranferred.
30        if (cnt < 4) then
31           if (DIN1 = DIN2) then
32              regOut(cnt) <= '0';
33           else
34              regOut(cnt) <= '1';
35           end if;
36           case (cnt) is
37              when 0 to 2 => EN <= '0';
38              when 3 => EN <= '1';
39              when others => null;
40           end case;
41           cnt <= cnt + 1;
42        --3) For 2 clock periods of time regOut content is tranferred to the output signals as follows:
43        --regOut(0) => DOUT1;
44        --regOut (1) => DOUT2;
45        --and so on...
46        --when cnt = 5 (the second clock period of 2) EN is set to '0' so that for the following 2 clock
47        --periods of time, output signals are kept in high-impedance state.
48        elsif (cnt = 4) then
49           cnt <= cnt + 1;
50        elsif (cnt = 5) then
51           EN <= '0';
52           cnt <= cnt + 1;
53        --4) For 2 clock periods of time output signals are kept to high-impedance state and during the last
54        --clock period cnt is set to 0 so that the entity can restart its operation:
55        elsif (cnt = 6) then
56           cnt <= cnt + 1;
57        elsif(cnt = 7) then
58           regOut <="ZZZZ";
59           cnt <= 0;
60        end if;
61     end if;
62  end process;
63  --5) output computation:
64  DOUT1 <= regOut (0) when (EN = '1') else 'Z';
65  DOUT2 <= regOut (1) when (EN = '1') else 'Z';
66  DOUT3 <= regOut (2) when (EN = '1') else 'Z';
67  DOUT4 <= regOut (3) when (EN = '1') else 'Z';
68  end Behavioral;
69  -- Commento: ok - piccola differenza d'interpretazione dei vincoli - comunque ok (29)
70
```