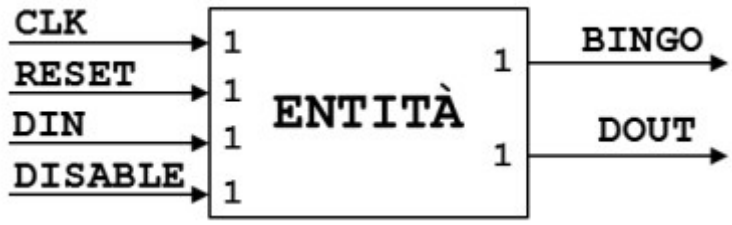
14 January 2015

Create a clock-synchronous VHDL entity which starting from a reset analyses a serial input **DIN**.

If the first 8 bits are the 1-complement to the second 8 bits (example: 00011111 – 11100000), the output **BINGO**, which is normally 0, will emit an impulse for 1 clock cycle to signal the start of the serial output stream **DOUT**. Immediately after the pulse of **BINGO**, **DOUT** must emit the first 8 bits arriving to **DIN**.

If the 1-complement condition is not met, **BINGO** must still emit its impulse but **DOUT** must remain in high impedance state.

When the **DISABLE** signal is active both outputs must be in high impedance state.



```
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   --According to the description, the entity functioning seems to be one-shot.
7
8   entity PF is
9      port(CLK, RST, DIN, DIS: in std_logic;
10          BINGO, DOUT: out std_logic);
11  end PF;
12
13  architecture Behavioral of PF is
14
15  signal reg1: std_logic_vector (2 downto 0):= "ZZZ";
16  signal reg2: std_logic_vector (0 to 2):= "ZZZ";
17  signal regD, regB: std_logic:='Z';
18  signal upl: bit:='0';
19  signal cnt: integer range 0 to 10:=0;
20
21  begin
22
23  process (CLK, RST)
24  begin
25
26  --After a reset occurs, the entity begins to 'read' a serial input (DIN): 16 bits.
27  --NB: I've programmed just 6 bits in order to obtain a short testbench;
28  --in order to have a versatile solution, a constant can be used, but the
29  --following source code must be modified.
30  if (RST = '1') then
31     cnt <= 0;
32     reg1 <= "ZZZ";
33     reg2 <= "ZZZ";
34     upl <= '0';
35
36  elsif (rising_edge(CLK)) then
37     --The first 8 bits (3) are forward-uploaded
38     --into a register, reg1:
39     if (cnt < 3) then
40        reg1 <= reg1(1 downto 0) & DIN;
41        regB <= '0';
42        regD <= 'Z';
43        cnt <= cnt + 1;
44     --The remaining 8 bits (3 or N) are reverse-uploaded into another
45     --register, reg2, in order to execute the subsequent comparison:
46     elsif (cnt > 2 and cnt < 5) then
47        reg2 <= DIN & reg2(0 to 1);
48        regB <= '0';
49        regD <= 'Z';
50        cnt <= cnt + 1;
51     --When the last bit is uploaded, the output signal 'BINGO'
52     --(normally at low level) goes high level for one clock period of time
53     --(regB is actually updated and then tranferred to BINGO, if the entity is
54     --enabled):
55     elsif (cnt = 5) then
56        reg2 <= DIN & reg2(0 to 1);
57        regB <= '1';
58        regD <= 'Z';
59        cnt <= cnt + 1;
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
```

```
77          --If the first 8 (3) bits and the last 8 (3) ones are specular-like
78          --the first 8 (3) bits are tranferred to the serial output 'DOUT'
79          --otherwise DOUT is kept in high impedance state.
80          --NB: since reg1 is forward-uploaded, it must be tranferred to DOUT
81          --from the MSB to the LSB (*);
82          --NB: the signal 'upl' is used so that it's not required to control again
83          --whether reg1=reg2 or not (**);
84          --NB: reg1 and reg2 cannot be compared before this point, since reg2 is not
85          --completely uploaded yet.
86          elsif (cnt = 6) then
87             regB <= '0';
88             cnt <= cnt + 1;
89             if (reg1 = reg2) then
90                upl <= '1';
91                regD <= reg1(2);  --(*)
92                reg1 <= reg1(1 downto 0) & 'Z';
93             else
94                regD <= 'Z';
95             end if;
96
97          --(**)
98          --NB: this 'elsif' body is necessary since as reg1 is tranferred to the output
99          --its content is updated and it can nomore be compared to reg2.
100         elsif (cnt = 7) then
101            regB <= '0';
102            if (upl = '1') then
103               regD <= reg1(2);  --(*)
104               reg1 <= reg1(1 downto 0) & 'Z';
105            else
106               regD <= 'Z';
107            end if;
108         --NB: after the first DIN 8 (3) bits are tranferred to DOUT, the latter
109         --is kept in high impedance state until a new reset occurs (one-shot).
110         end if;
111      end if;
112      end process;
113
114      --If the entity is not disabled (DIS = 1), regD and regB are respectively
115      --tranferred to DOUT and BINGO.
116      DOUT <= regD when (DIS = '0') else 'Z';
117      BINGO <= regB when (DIS = '0') else 'Z';
118
119      end Behavioral;
```