14 April 2015

Create a clock-synchronous VHDL entity which starting from a reset samples the inputs **DIN1** and **DIN2** and **DIN3**.

The entity must count the number of 1 arriving at the inputs and generate 3 corresponding outputs **DOUT1**, **DOUT2**, and **DOUT3** with the following behaviour:

- Initially, the outputs must be in high impedance state. Then the output correspondent to the input with the highest count of 1 must be set to 1 while the other outputs must be set to 0.
- If 2 inputs have the same counts, the correspondent outputs must be set to high impedance and the remaining output must be set to 1 if its correspondent input has higher count than the other 2 or 0 otherwise.
- If all inputs have the same count, all the 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   --Post-commentario (in italiano, eccezionalmente):
7   --questa soluzione NON è testata in quanto mi è scaduta la licenza per Modelsim
8   --e non avendone più bisogno, non mi va di scaricarlo di nuovo. In teoria, dovrebbe funzionare
9   --ma potrei ricordare male il funzionamento delle variabili e inoltre il testo di per se non è
10  --molto chiaro. Sintatticamente è corretto.
11
12  --Dal testo, pare che il funzionamento sia in tempo reale ed è per questo che preferisco usare
13  --le variabili piuttosto che dei segnali di controllo, per non avere il ritardo sul clock
14  --nell'aggiornamento dei segnali e quindi sulle uscite (non ricordo se le uscite portano
15  --comunque un ritardo in realtà); è da tenere conto comunque che i valori delle variabili non
16  --sono controllabili in simulazione e che quindi è più difficile trovare eventuali bug.
17  --Usando le variabili evito di dover implementare due processi distinti (ammesso che funzioni).
18
19  entity CNT_1 is
20     port (CLK, DIN1, DIN2, DIN3, RESET: in std_logic;
21           DOUT1, DOUT2, DOUT3: out std_logic);
22  end CNT_1;
23
24  architecture Behavioral of CNT_1 is
25
26  begin
27
28     process (RESET, CLK)
29
30     variable cnt1: integer :=0;
31     variable cnt2: integer :=0;
32     variable cnt3: integer :=0;
33
34     begin
35
36        --General reset:
37        if (RESET = '1') then
38           DOUT1 <= 'Z';
39           DOUT2 <= 'Z';
40           DOUT3 <= 'Z';
41           cnt1 := 0;
42           cnt2 := 0;
43           cnt3 := 0;
44
45        --when RESET = 0:
46        elsif (rising_edge(CLK)) then
47
48           --Counting input 1-pulses:
49           if (DIN1 = '1') then
50              cnt1 := cnt1 + 1;
51           end if;
52           if (DIN2 = '1') then
53              cnt2 := cnt2 + 1;
54           end if;
55           if (DIN3 = '1') then
56              cnt3 := cnt3 + 1;
57           end if;
58
59           --Comparing:
60           --All of counter values are equal:
61           if (cnt1 = cnt2 and cnt2 = cnt3) then
62              DOUT1 <= 'Z';
63              DOUT2 <= 'Z';
64              DOUT3 <= 'Z';
65           --Since this is an alternative condition, surely cnt3 value
66           --differs from the one of cnt1 and cnt2.
67           elsif (cnt1 = cnt2 ) then
68              if (cnt1 > cnt3) then
69                 DOUT1 <= 'Z';
70                 DOUT2 <= 'Z';
71                 DOUT3 <= '0';
72              else
73                 DOUT1 <= 'Z';
74                 DOUT2 <= 'Z';
75                 DOUT3 <= '1';
76              end if;
```

```
77              --cnt2 value is surely different
78              elsif (cnt1 = cnt3) then
79                 if (cnt1 > cnt2) then
80                    DOUT1 <= 'Z';
81                    DOUT2 <= '0';
82                    DOUT3 <= 'Z';
83                 else
84                    DOUT1 <= 'Z';
85                    DOUT2 <= '1';
86                    DOUT3 <= 'Z';
87                 end if;
88              --cnt1 value is surely different
89              elsif (cnt2 = cnt3) then
90                 if (cnt1 > cnt2) then
91                    DOUT1 <= '1';
92                    DOUT2 <= 'Z';
93                    DOUT3 <= 'Z';
94                 else
95                    DOUT1 <= '0';
96                    DOUT2 <= 'Z';
97                    DOUT3 <= 'Z';
98                 end if;
99              ---------------------------------------
100             --At this point, SURELY all of the counters values
101             --differs from each other.
102             elsif (cnt1 < cnt2) then
103                if (cnt2 < cnt3) then  --cnt1 < cnt2 < cnt3
104                   DOUT1 <= '0';
105                   DOUT2 <= '0';
106                   DOUT3 <= '1';
107                else                   --cnt2 > cnt1 AND cnt2 > cnt3 => cnt2 is the maximum!
108                   DOUT1 <= '0';
109                   DOUT2 <= '1';
110                   DOUT3 <= '0';
111                end if;
112             --cnt1 > cnt2 is valid because it's an alternative to the condition right above
113             elsif (cnt3 < cnt2) then  --cnt1 > cnt2 > cnt3
114                DOUT1 <= '1';
115                DOUT2 <= '0';
116                DOUT3 <= '0';
117             --cnt1 > cnt2 AND cnt3 > cnt2 => cnt2 is the minimum!
118             --Hence, I'll compare just cnt1 and cnt3.
119             elsif (cnt3 < cnt1) then
120                DOUT1 <= '1';
121                DOUT2 <= '0';
122                DOUT3 <= '0';
123             else
124                DOUT1 <= '0';
125                DOUT2 <= '0';
126                DOUT3 <= '1';
127             end if;
128          end if;
129       end process;
130
131    end Behavioral;
132
```