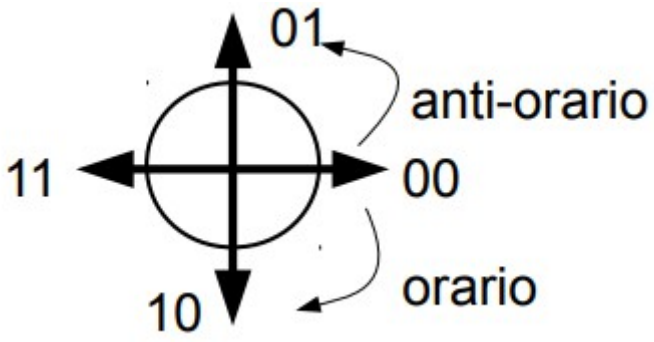
9 September 2013

Create a clock-synchronous VHDL entity which implements the driving signals of a step-by-step motor.

The entity receives pairs of bits as input **DIN** indicating the direction of the motor axis according to the following figure:



Translation notes:
anti-orario = Counter clockwise
orario = clockwise

If the input receives a sequence of bits describing a complete rotation of the motor clockwise, the entity must emit an impulse **P_OR** for half a clock cycle. Similarly, a complete rotation counter clockwise must cause an output impulse **P_AN**. If the motor rotates 10 times clockwise (counter clockwise) the output signal **P_OR10** (**P_AN10**) must be activated and kept active until the motor rotates in the opposite direction.

The enable signal **EN** determines the absence of control. In this case both **P_OR** and **P_AN** must be active at the same time.

```
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   entity StepToStep is
7      port (CLK, EN: in std_logic;
8            DIN: in std_logic_vector (1 downto 0);
9            POR, PANT, POR10, PANT10: out std_logic);
10  end StepToStep;
11
12  --In this version it is supposed that the entity starts
13  --from "00" so at the beginning, the value of DIN must be equal to
14  --"00"; after this value occurs, the entity starts counting the revolutions.
15
16  architecture Behavioral of StepToStep is
17
18  --cntOR, cntANT: for counting revolutions;
19  --REG: to store the occurences of DIN and control the right occurrence succession;
20  --regOR, regANT, regOR10, regANT10: for computing the output.
21  signal cntOR, cntANT: integer range 0 to 10:=0;
22  signal REG: std_logic_vector (7 downto 0):=(others => '0');
23  signal regOR, regANT, regOR10, regANT10: std_logic:='0';
24  signal cnt: integer range 0 to 10:=0;
25
26  --According to the request, 10 revolutions should be counted, but the testbench
27  --will be too long and confused. It is enough to set the value of FS (Full Scale) to 10
28  --to fulfill the required purpose.
29  constant FS: integer:=3;
30
31  begin
32
33     process (CLK, EN)
34     begin
35        --Entity disabled:
36        if (EN = '0') then
37           regOR <= '1';
38           regANT <= '1';
39        --Entity enabled:
40        elsif rising_edge (CLK) then
41           --does nothing until "00" occurs...
42           if ((DIN /= "00") and (cnt = 0)) then
43              regOR <= '0';
44              regANT <= '0';
45           --when "00" occurs, it goes on...
46           elsif ((DIN = "00") and (cnt = 0)) then
47              regOR <= '0';
48              regANT <= '0';
49              cnt <= cnt + 1;
50           ----------------------------------------------------------------
51           --four instances of DIN are stored into REG...
52           elsif (cnt > 0) then
53              case cnt is
54                 when 1 => REG (7 downto 6) <= DIN;
55                           regOR <= '0';
56                           regANT <= '0';
57                           cnt <= cnt + 1;
58                 when 2 => REG (5 downto 4) <= DIN;
59                           regOR <= '0';
60                           regANT <= '0';
61                           cnt <= cnt + 1;
62                 when 3 => REG (3 downto 2) <= DIN;
63                           regOR <= '0';
64                           regANT <= '0';
65                           cnt <= cnt + 1;
66                 when 4 => REG (1 downto 0) <= DIN;
67                           regOR <= '0';
68                           regANT <= '0';
69                           cnt <= cnt + 1;
70                 when others => null;
71              end case;
72              --NOTE: output has been kept at low level so far...
73              ------------------------------------------------------------------------
74
75
76
```

```
77              --The storage of the instances of DIN is complete; the sequence of them can be
78              --contolled now:
79              if (cnt = 5) then
80                 --If a clockwise revolution occurs* it is counted unless 10 clockwise revolutions
81                 --have already been counted, in this case it is not counted since the full scale
82                 --is reached.
83                 if ((REG = "10110100") and (cntOR < FS)) then
84                    if (cntOR = (FS -1)) then
85                       regOR10 <= '1';
86                    end if;
87                    cntOR <= cntOR + 1;
88                    cnt <= 2;
89                 elsif ((REG = "10110100") and (cntOR = FS)) then
90                    cnt <= 2;
91
92                 --Analogous operations are executed when an anticlockwise** revolution occurs.
93                 elsif ((REG = "01111000") and (cntANT < FS)) then
94                    if (cntANT = (FS - 1)) then
95                       regANT10 <= '1';
96                    end if;
97                    cntANT <= cntANT + 1;
98                    cnt <= 2;
99                 elsif ((REG = "01111000") and (cntANT = FS)) then
100                   cnt <= 2;
101
102                --If the values of DIN don't follow a succession which corresponds
103                --to a revolution, nothing is done, plus, if the last instance
104                --of DIN has not been "00", the entity restarts from the beginning (line 42);
105                --otherwise, if the last instance od DIN has been "00", the entity can restart
106                --from line 58, since it is as if the first required instance of DIN has occurred
107
108                --NOTE: all the previous cases exclude each other, and this is correct!***
109                --NOTE: in any case before line 111, the entity restarts from line 58, since it is
110                --as if the required instance of DIN has occurred for the next revolution.
111                elsif (REG (1 downto 0) /= "00") then
112                   if (DIN = "00") then
113                      cnt <= 1;
114                   else
115                      cnt <= 0;
116                   end if;
117                elsif (REG (1 downto 0) = "00") then
118                   cnt <= 2;
119                end if;
120

121                ---------------------------------------------------------------------------
122                --If 10 clockwise (anticlockwise) revolutions have been counted and an
123                --anticlockwise (clockwise) revolution occurs, cntOR (cntANT) is set to 0
124                --so that the number of revolutions can be counted again from 0
125                --and regOR (regANT) is set to '0' so that the output can go back to low
126                --level, as it is required.
127                --***NOTE: this section is not an elsif of the previous if contruct because it is
128                --not an alternative set of operations among the previous ones, it must be
129                --executed in any case!
130                if ((REG = "10110100") and (cntANT = FS)) then
131                   cntANT <= 0;
132                   regANT10 <= '0';
133                elsif ((REG = "01111000") and (cntOR = FS)) then
134                   cntOR <= 0;
135                   regOR10 <= '0';
136                end if;
137
138                --In any case, when the succession of the occurences of DIN is controlled, the
139                --occurrence of DIN for the next revolution must be stored, because the entity
140                --restarts from line 58 where the second occurrence of DIN (the first one is
141                --the final "00" of the previous revolution which has been completed) should
142                --has been already stored.
143                REG (7 downto 6) <= DIN;
144

145             end if;
146          end if;
147
148       end if;
149
150    end process;
```

```
151
152     --When a clockwise revolution* is complete, the clock signal is sent to output so that it can
153     --go to high level for half a clock period; otherwise, the output is set to regOR that is set
154     --to '0' until the entity is disabled (when the entity is enabled regOR and regANT both are set
155     --to '1'). Analogous operation for an anticlockwise revolution**.
156     POR <= CLK when ((REG = "10110100") and (cnt = 5)) else regOR;
157     PANT <= CLK when ((REG = "01111000") and (cnt = 5)) else regANT;
158
159     --POR10 and PORANT are simply set to regOR10 and regANT10 respectively.

160     POR10 <= regOR10;
161     PANT10 <= regANT10;
162
163     end Behavioral;
```