10 November 2008

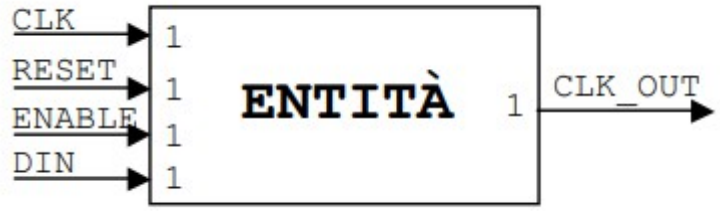
Create a clock-synchronous VHDL entity which generates a new clock signal **CLK_OUT** with configurable frequency and duty cycle.

Starting from a reset, a serial input **DIN** is read to acquire a 5-bit BCD-coded word which determines the characteristics of **CLK_OUT** according to the following table:

Valore di DIN	12	18	20	23	28	31
Frequenza di CLK_OUT	CLK/4	CLK/2	CLK/4	CLK/8	CLK/4	CLK/8
Duty-cycle di CLK_OUT	25%	50%	50%	50%	75%	75%

When the value read from **DIN** doesn't match any of the values in the table, **CLK_OUT** is equal to 1.

The **ENABLE** signal deactivate the output by setting it to high impedance while the entity's internal function is not affected.



```
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 CLKGEN is
7      port (CLK, RST, EN, DIN: in std_logic;
8            CLK_OUT: out std_logic);
9   end CLKGEN;
10
11  architecture Behavioral of CLKGEN is
12
13  --cnt: to scan the input;
14  --cnt1: to display the output;
15  --regOUT: to store what must be sent to output;
16  --regIN: To store the input.
17  signal cnt: integer range 0 to 10:=0;
18  signal cnt1: integer range 0 to 20:=0;
19  signal regOUT: std_logic:='Z';
20  signal regIN: std_logic_vector (4 downto 0):="00000";
21
22  begin
23
24     process (CLK, RST, EN)
25     begin
26        if (RST = '1') then
27           cnt <= 0;
28           cnt1 <= 0;
29           regIN <= "00000";
30           regOUT <= 'Z';
31        elsif rising_edge (CLK) then
32           if (cnt < 5) then
33              regIN <= regIN (3 downto 0) & DIN;
34              cnt <= cnt + 1;
35              regOUT <= 'Z';
36
37           elsif (cnt = 5) then
38              case regIN is
39                 --12: Period = 8 clock pulses;
40                 --output is high for 2 clock pulses:
41                 when "01100" => if (cnt1 < 2) then
42                                   regOUT <= '1';
43                                   cnt1 <= cnt1 + 1;
44                                 elsif ((cnt1 >= 2) and (cnt1 < 7)) then
45                                   regOut <= '0';
46                                   cnt1 <= cnt1 + 1;
47                                 elsif (cnt1 = 7) then
48                                   regOUT <= '0';
49                                   cnt1 <= 0;
50                                   --In this way, the "loop" can be repeated.
51                                 end if;
52
53                 --18: Period = 4 clock pulses;
54                 --output is high for 2 clock pulses:
55                 when "10010" => if (cnt1 < 2) then
56                                   regOUT <= '1';
57                                   cnt1 <= cnt1 + 1;
58                                 elsif ((cnt1 >= 2) and (cnt1 < 3)) then
59                                   regOut <= '0';
60                                   cnt1 <= cnt1 + 1;
61                                 elsif (cnt1 = 3) then
62                                   regOUT <= '0';
63                                   cnt1 <= 0;
64                                   --In this way, the "loop" can be repeated.
65                                 end if;
66
67
68
69
70
71
72
73
74
75
76
```

```
77                --20: Period = 8 clock pulses;
78                --output is high for 4 clock pulses:
79                when "10100" => if (cnt1 < 4) then
80                                  regOUT <= '1';
81                                  cnt1 <= cnt1 + 1;
82                                elsif ((cnt1 >= 3) and (cnt1 < 7)) then
83                                  regOut <= '0';
84                                  cnt1 <= cnt1 + 1;
85                                elsif (cnt1 = 7) then
86                                  regOUT <= '0';
87                                  cnt1 <= 0;
88                                  --In this way, the "loop" can be repeated.
89                                end if;
90
91                --23 Period = 16 clock pulses;
92                --output is high for 8 clock pulses:
93                when "10111" => if (cnt1 < 8) then
94                                  regOUT <= '1';
95                                  cnt1 <= cnt1 + 1;
96                                elsif ((cnt1 >= 7) and (cnt1 < 15)) then
97                                  regOut <= '0';
98                                  cnt1 <= cnt1 + 1;
99                                elsif (cnt1 = 15) then
100                                 regOUT <= '0';
101                                 cnt1 <= 0;
102                                 --In this way, the "loop" can be repeated.
103                               end if;
104
105               --28: Period = 8 clock pulses;
106               --output is high for 6 clock pulses:
107               when "11100" => if (cnt1 < 6) then
108                                 regOUT <= '1';
109                                 cnt1 <= cnt1 + 1;
110                               elsif ((cnt1 >= 5) and (cnt1 < 7)) then
111                                 regOut <= '0';
112                                 cnt1 <= cnt1 + 1;
113                               elsif (cnt1 = 7) then
114                                 regOUT <= '0';
115                                 cnt1 <= 0;
116                                 --In this way, the "loop" can be repeated.
117                               end if;
118
119               --31: Period = 16 clock pulses;
120               --output is high for 12 clock pulses;
121               when "11111" => if (cnt1 < 12) then
122                                 regOUT <= '1';
123                                 cnt1 <= cnt1 + 1;
124                               elsif ((cnt1 >= 11) and (cnt1 < 15)) then
125                                 regOut <= '0';
126                                 cnt1 <= cnt1 + 1;
127                               elsif (cnt1 = 15) then
128                                 regOUT <= '0';
129                                 cnt1 <= 0;
130                                 --In this way, the "loop" can be repeated.
131                               end if;
132               when others => regOUT <= 'Z';
133            end case;
134
135         end if;
136      end if;
137   end process;
138
139   CLK_OUT <= regOUT when (EN = '1') else 'Z';
140
141   end Behavioral;
142
```