12 September 2008

Create a clock-synchronous VHDL entity which implements a generator of 4 signals with programmable duty-cycle.

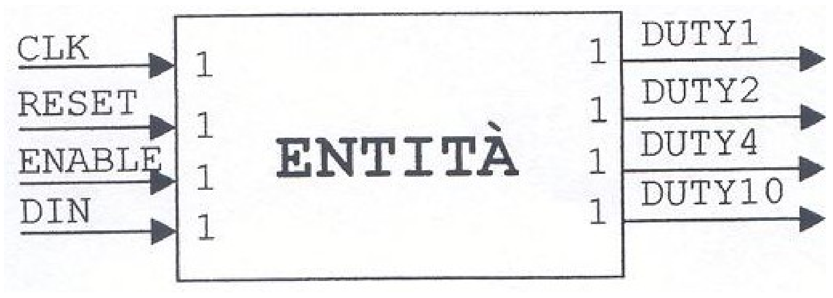
Starting from a reset, the entity receives 3 bits at a serial input **DIN**, these 3 bits determine the duty-cycle of the output signals according to the following table:

Value	0	1	2	3	4	5	6	7
Duty-cycle	20%	30%	40%	50%	60%	70%	80%	90%

The four output signals (**DUTY1**, **DUTY2**, **DUTY4**, **DUTY10**) produce signals with the same duty-cycle and frequencies equal to:

DUTY2 : F(DUTY1) / 2
DUTY4 : F(DUTY1) / 4
DUTY10 : F(DUTY1) / 10

Finally, when **ENABLE** is 0, outputs are at high impedance.



```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity duty_cycle is
   port(RST, CLK, DIN, EN: in std_logic;
        DUTY1, DUTY2, DUTY4, DUTY10: out std_logic);
end duty_cycle;


architecture Behavioral of duty_cycle is
   --cnt: for the input;
   --regIN: to store the value for computing the duty cycle;
   --DC: for converting regIN to integer.
   signal cnt : integer range 0 to 5:=0;
   signal regIN: std_logic_vector(2 downto 0):="000";
   signal DC: integer range 0 to 7:=0;

   signal cnt1: integer range 0 to 10:=0;    --For the output: (1)
   signal cnt2: integer range 0 to 20:=0;    --(2)
   signal cnt4: integer range 0 to 40:=0;    --(3)
   signal cnt10: integer range 0 to 100:=0;  --(4)

   begin

      process (CLK, RST, EN)
      begin
         if (RST = '1')then
            --General reset:
            cnt <= 0;
            regIN <= "000";
            cnt <= 0;
            cnt1 <= 0;
            cnt2 <= 0;
            cnt4 <= 0;
            cnt10 <= 0;
            DC <= 0;
         elsif rising_edge(CLK) then
            --Input storing:
            ------------------------------------------------------
            case cnt is
               when 0 to 2 => regIN <= regIN (1 downto 0) & DIN;
                              cnt <= cnt + 1;
                              DUTY1 <= 'Z';
                              DUTY2 <= 'Z';
                              DUTY4 <= 'Z';
                              DUTY10 <= 'Z';
               when 3 => DC <= conv_integer(regIN);
                         cnt <= cnt + 1;
               when others => null;
            end case;

            -------------------------------------------------------
            --Output computation (next page):
            --A "loop" for each output line is executed;
            --A minimum value of 10 clock pulses for the period has been chosen (DUTY1);
            --The other values are:
            --20 pulses (DUTY2);
            --40 pulses (DUTY4);
            --100 pulses (DUTY10).
            --For each "loop" a specific counter signal has been used and this one is set to zero
            --when it reaches the maximum value (10 for cnt1, 20 for cnt2 and so on), so that the
            --output bitstream can be generated indefinitely.
            ---------------------------------------------------------
            if ((EN = '1') and (cnt > 3)) then
               --Modify using a procedure!!!
               --parameters: max value of cnt#; DC; cnt#
               --DUTY1:
               if (cnt1 < 10) then
                  if ((cnt1 >= 0) and  (cnt1 < 2+DC)) then
                     DUTY1 <= '1';
                     cnt1 <= cnt1 + 1;
                  elsif ((cnt1 >= 2+DC) and (cnt1 <= 8)) then
                     DUTY1 <= '0';
                     cnt1 <= cnt1 + 1;
                  elsif (cnt1 = 9) then
                     DUTY1 <= '0';
                     cnt1 <= 0;
                  end if;
               end if;

               --DUTY2:
               if (cnt2 < 20) then
                  if ((cnt2 >= 0) and  (cnt2 < 2*(2+DC))) then
                     DUTY2 <= '1';
                     cnt2 <= cnt2 + 1;
                  elsif ((cnt2 >= 2*(2+DC)) and (cnt2 <= 18)) then
                     DUTY2 <= '0';
                     cnt2 <= cnt2 + 1;
                  elsif (cnt2 = 19) then
                     DUTY2 <= '0';
                     cnt2 <= 0;
                  end if;
               end if;

               --DUTY4:
               if (cnt4 < 40) then
                  if ((cnt4 >= 0) and  (cnt4 < 4*(2+DC))) then
                     DUTY4 <= '1';
                     cnt4 <= cnt4 + 1;
                  elsif ((cnt4 >= 4*(2+DC)) and (cnt4 <= 38)) then
                     DUTY4 <= '0';
                     cnt4 <= cnt4 + 1;
                  elsif (cnt4 = 39) then
                     DUTY4 <= '0';
                     cnt4 <= 0;
                  end if;
               end if;

               --DUTY10:
               if (cnt10 < 100) then
                  if ((cnt10 >= 0) and  (cnt10 < 10*(2+DC))) then
                     DUTY10 <= '1';
                     cnt10 <= CNT10 + 1;
                  elsif ((cnt10 >= 10*(2+DC)) and (cnt10 <= 108)) then
                     DUTY10 <= '0';
                     cnt10 <= CNT10 + 1;
                  elsif (cnt10 = 109) then
                     DUTY10 <= '0';
                     cnt10 <= 0;
                  end if;
               end if;

            else
               --Entity disabled:
               DUTY1 <= 'Z';
               DUTY2 <= 'Z';
               DUTY4 <= 'Z';
               DUTY10 <= 'Z';
            end if;
         end if;

      end process;

end Behavioral;
```