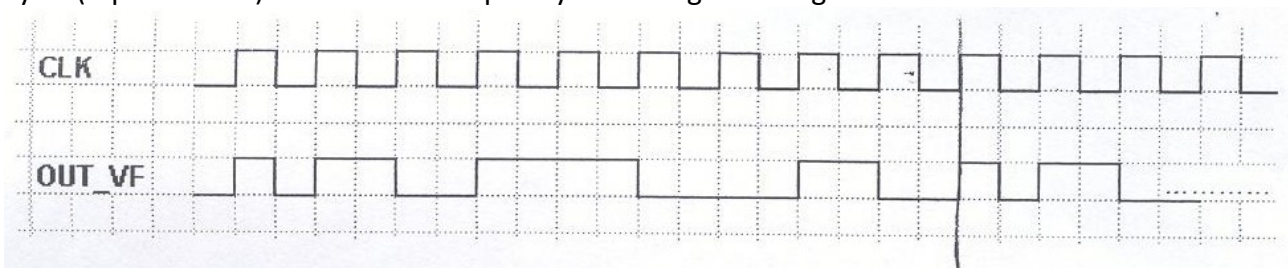
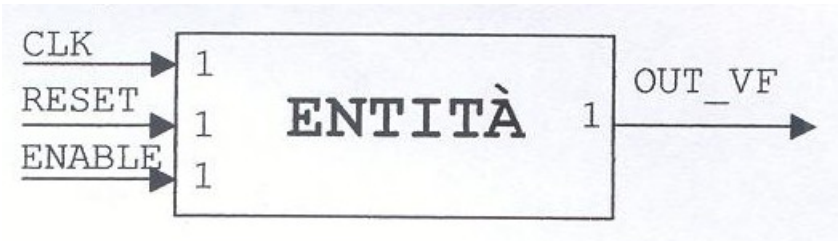
14 October 2008

Create a clock-synchronous VHDL entity which generates a signal **OUT_VF** with constant duty cycle (equal to 50%) and variable frequency according to the figure below.



Starting from a reset, this output remains unchanged indefinitely.

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



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

entity freqGen is
   port (CLK, RST, EN: std_logic;
         OUT_VF: out std_logic);
end freqGen;

architecture Behavioral of freqGen is

   --cnt, cnt1: for entity execution;
   --E_B: to distinguish the passage to a value of frequency that repeats;
   --regOUT: for working also when the entity is disabled.
   signal cnt, cnt1: integer range 0 to 10:=0;
   signal E_B: std_logic:='0';
   signal regOUT: std_logic;

   begin
      process (CLK, RST, EN)
      begin
         if (RST = '1') then
            cnt <= 0;
            cnt1 <= 1;
            E_B = '0'
         elsif rising_edge (CLK) then
            --It may be implemented in a more simple way: a case construction in which the
            --output is computed as shown in the image (see the text) for each case...
            --but this is ridicolous.
            --This solution is more difficult to undestand, but it's funnier and shorter:
            --cnt1 tells the number of clock pulses in a semiperiod; cnt is needed to compute
            --a single period; for the first semiperiod (half count) regOUT is set to high level
            --in the second semiperiod regOUT is set to low level.
            if (cnt < cnt1) then
               regOUT <= '1';
               cnt <= cnt + 1;
            elsif ((cnt >= cnt1) and (cnt < 2*cnt1)) then
               regOUT <= '0';
               cnt <= cnt + 1;
            end if;

            --The following constructions are required to change the frequency:
            --NOTE: after the frequency assumes the smallest value, it rises again;
            --to undestant whether the frequency is increasing or decreasing, E_B is used;
            --It is set to 1 when the frequency is increasing;
            --it is set to 0 when the frequency is decreasing:
            if ((cnt = 1) and (cnt1 = 1)) then
               cnt1 <= 2;
               cnt <= 0;
            end if;

            if ((cnt = 3) and (cnt1 = 2) and (E_B = '0')) then
               cnt1 <= 4;
               cnt <= 0;
               E_B <= '1';
            end if;

            if (cnt = 7 and cnt1 = 4) then
               cnt1 <= 2;
               cnt <= 0;
            end if;

            if (cnt = 3 and cnt1 = 2 and E_B = '1') then
               cnt1 <= 1;
               cnt <= 0;
               E_B <= '0';
            end if;
         end if;
      end process;

--Output is displayed only when the entity is enabled, otherwise is set to
--high impedance:
OUT_VF <= regOUT when (EN = '1') else 'Z';

end Behavioral;
```