```
-- Contador de modulo programable entre 2 y 1023


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity cnt_prog is
port(
      clk,nrst: in std_logic;
      modulo      : in std_logic_vector(9 downto 0);
      FCD   : buffer std_logic
);

end entity;

architecture rtl of cnt_prog is

      signal Q: std_logic_vector(9 downto 0);

begin

process(clk,nrst)
begin
   if nrst = '0' then
      Q <= (others => '0');
   elsif clk'event and clk = '1' then
      if FCD = '1' then
       Q <= (0 => '1', others =>'0');
      else
       Q <= Q + 1;
      end if;
   end if;
end process;

FCD <=  '1' when modulo <= Q else
        '0';

end rtl;
```

```
-- Test-bench del contador de modulo programable entre 2 y 1023


library ieee;
use ieee.std_logic_1164.all;

entity tb_cnt_prog is
end entity;

architecture test of tb_cnt_prog is

      signal clk, nrst: std_logic;
```

```
        signal FCD: std_logic;
        signal modulo: std_logic_vector(9 downto 0);

        constant T_CLK : time := 1 us;
        signal final: boolean := false;

begin
dut: entity Work.cnt_prog(rtl)
port map(
        clk => clk,
        nrst => nrst,
        FCD => FCD,
        modulo => modulo
        );

process
begin

        clk <= '0';
        wait for T_CLK/2;
        clk <= '1';
        wait for T_CLK/2;
          if final then
             wait;
          end if;
end process;

process
begin
        -- Inicializacion asincrona
        nrst <= '0';
        modulo <= (others => '1');
        wait until clk'event and clk = '1';
        wait until clk'event and clk = '1';
        nrst <= '1';
        wait until clk'event and clk = '1';

        -- Probamos con el modulo maximo
        modulo <= (others => '1');    -- 1023
        wait for 1100*T_CLK;
        wait until clk'event and clk = '1';

        -- Probamos con modulo minimo
        modulo <= "0000000010"; -- 2
        wait for 10*T_CLK;
        wait until clk'event and clk = '1';

        -- Probamos con modulo intermedio
        modulo <= "0000010001"; -- 17
        wait for 25*T_CLK;
        wait until clk'event and clk = '1';

        -- Probamos con modulo 1 (la salida tiene que ser 1)
        modulo <= "0000000001"; -- 1
        wait for 5*T_CLK;
        wait until clk'event and clk = '1';
```

```
        modulo <= "1000001000"; -- 520
        wait for 550*T_CLK;
        wait until clk'event and clk = '1';

        -- Probamos con modulo 0 (salida tiene que ser 1)
        modulo <= "0000000000";
        wait for 5*T_CLK;
        wait until clk'event and clk = '1';

        final <= true;
        wait;
end process;

end test;
```