```
-- Modelo para el Cronometro del ejercicio 5
-- Tiene errores sintácticos que hay que corregir
-- También tiene 3 errores funcionales que se pueden corregir por
separado


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

entity cronometro is port(
      clk       : in std_logic;
      rst_n     : in std_logic;
       sto_1    : in std_logic;
      sto_2     : in std_logic;
      ini_c     : in std_logic;
      seg_u     : buffer std_logic_vector(3 downto 0);
      seg_d     : buffer std_logic_vector(3 downto 0);
      dec       : buffer std_logic_vector(3 downto 0);
      seg_u_st1 : buffer std_logic_vector(3 downto 0);
      seg_d_st1 : buffer std_logic_vector(3 downto 0);
      dec_st1   : buffer std_logic_vector(3 downto 0);
      seg_u_st2 : buffer std_logic_vector(3 downto 0);
      seg_d_st2 : buffer std_logic_vector(3 downto 0);
      dec_st2   : buffer std_logic_vector(3 downto 0)
);
end entity;

architecture rtl of cronometro is
      signal cnt       : std_logic_vector(22 downto 0);
      signal tic       : std_logic;
      signal ena       : std_logic;
      signal eoc_dec   : std_logic;
      signal eoc_seg_u : std_logic;
      signal eoc_seg_d : std_logic;
      signal reset     : std_logic;


-- Para diseño físico descomentar la línea 47 y comentar la 48
-- constant T_DEC : integer := 5000000; -- para diseño físico (con reloj
de 50 MHz)
constant T_DEC : integer := 5;-- para simulación
begin


 -- Subsistema_1  --> Timer de modulo 5
 process(clk, rst_n)
 begin
   if rst_n = '0' then
     cnt <= (0 => '1',others => '0');
   elsif clk'event and clk = '1' then
     if tic = '1' or reset = '1' then
       cnt <= (0 => '1',others => '0');
     else
       cnt <= cnt + 1;
     end if;
```

```
    end if;
  end process;

  tic <= '1' when cnt = T_DEC else '0';
  -- Para sincronizar la cuenta con inic_c
  reset <= '1' when ini_c = '1' and ena = '1' else '0';
  -- Fin subsistema_1

  -- Subsistema_2
  process(clk, rst_n)
  begin
   if rst_n = '0' then
     ena <= '0';
   elsif clk'event and clk = '1' then
     if ini_c = '1' then
       ena <= '1';
     elsif (seg_d&seg_u&dec) = X"599" and tic = '1' then
       ena <= '0';
     end if;
   end if;
  end process;
  -- Fin subsistema_2


  -- Subsistema_3  --> Contadores BCD de modulo 10 para unidades, decenas
 y centenas

 process(clk, rst_n)
  begin
    if rst_n = '0' then
      dec <= (others => '0');
    elsif clk'event and clk = '1' then
     if reset = '1' then                        -- Faltaba reset
         dec <= (others => '0');
      elsif  ena = '1' and tic = '1' then
        if eoc_dec = '1' then
          dec <= (others => '0');
        else
          dec <= dec + 1;
        end if;
      end if;
    end if;
  end process;

  eoc_dec <= '1' when ena = '1' and dec = 9 else '0';

  process(clk, rst_n)
   begin
     if rst_n = '0' then
       seg_u <= (others => '0');
     elsif clk'event and clk = '1' then
     if reset = '1' then                        -- Faltaba reset
         seg_u <= (others => '0');
      elsif eoc_dec = '1' and tic = '1' then    -- Fallo eoc_dec <= '1' --
 > eoc_dec = '1'
        if eoc_seg_u = '1' then
          seg_u <= (others => '0');
        else
          seg_u <= seg_u + 1;
        end if;
       end if;
     end if;
   end process;

  eoc_seg_u <= '1' when eoc_dec = '1' and seg_u = 9 else '0';

  process(clk, rst_n)
   begin
     if rst_n = '0' then
       seg_d <= (others => '0');
     elsif clk'event and clk = '1' then
       if reset = '1' then                       -- Faltaba reset
          seg_d <= (others => '0');
       elsif eoc_seg_u = '1' and tic = '1' then
        if eoc_seg_d = '1' then
          seg_d <= (others => '0');
        else
          seg_d <= seg_d + 1;
        end if;
       end if;
     end if;
   end process;

  eoc_seg_d <= '1' when eoc_seg_u = '1' and seg_d = 5 else '0';   --
 Fallo, contaba hasta seg_d = 9

 -- Fin subsistema_3

 -- Subsistema_4

  process(clk, rst_n)
  begin
    if rst_n = '0' then
     seg_u_st1 <= (others => '0');
     seg_d_st1 <= (others => '0');
     dec_st1   <= (others => '0');
     seg_u_st2 <= (others => '0');
     seg_d_st2 <= (others => '0');
     dec_st2   <= (others => '0');
    elsif clk'event and clk = '1' then
      if sto_1 = '1' then
       seg_u_st1 <= seg_u;
       seg_d_st1 <= seg_d;
       dec_st1   <= dec;
      end if;
      if sto_2 = '1' then
       seg_u_st2 <= seg_u;
       seg_d_st2 <= seg_d;
       dec_st2   <= dec;
      end if;
    end if;
  end process;

 -- Fin subsistema_4
```

```vhdl
end rtl;
```

```vhdl
-- Test del cronometro del ejercicio 5
--



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


entity test_cronometro is
end entity;

architecture test of test_cronometro is
 signal clk       : std_logic;
 signal rst_n     : std_logic;
 signal sto_1     : std_logic;
 signal sto_2     : std_logic;
 signal ini_c     : std_logic;
 signal seg_u     : std_logic_vector(3 downto 0);
 signal seg_d     : std_logic_vector(3 downto 0);
 signal dec       : std_logic_vector(3 downto 0);
 signal seg_u_st1 : std_logic_vector(3 downto 0);
 signal seg_d_st1 : std_logic_vector(3 downto 0);
 signal dec_st1   : std_logic_vector(3 downto 0);
 signal seg_u_st2 : std_logic_vector(3 downto 0);
 signal seg_d_st2 : std_logic_vector(3 downto 0);
 signal dec_st2   : std_logic_vector(3 downto 0);

 constant T_CLK : time := 20 ns;
 signal end_sim : boolean := false;
begin

 dut: entity work.cronometro(rtl) port map(
  clk       => clk,
  rst_n     => rst_n,
  sto_1     => sto_1,
  sto_2     => sto_2,
  ini_c     => ini_c,
  seg_u     => seg_u,
  seg_d     => seg_d,
  dec       => dec,
  seg_u_st1 => seg_u_st1,
  seg_d_st1 => seg_d_st1,
  dec_st1   => dec_st1,
  seg_u_st2 => seg_u_st2,
  seg_d_st2 => seg_d_st2,
  dec_st2   => dec_st2
 );

 process
 begin
```

```
    clk <= '0';
    wait for T_CLK/2;
    clk <= '1';
      if end_sim = true then
        wait;
      end if;
    wait for T_CLK/2;
  end process;
  process
  begin
    rst_n <= '0';
    sto_1 <= '0';
    sto_2 <= '0';
    ini_c <= '0';
    wait until clk'event and clk = '1';
    wait until clk'event and clk = '1';
    rst_n <= '1';
    wait until clk'event and clk = '1';
    -- Entradas inactivas, las salidas deben permanecer sin cambios
    wait for 12*T_CLK;
    wait until clk'event and clk = '1';
    -- Pulso de inicio
    ini_c <= '1';
    wait until clk'event and clk = '1';
    ini_c <= '0';

    -- Esperamos a que se realice una vuelta completa del cronometro
(59.9)
    wait until clk'event and clk = '1';
    wait for 3000*T_CLK;
    wait until clk'event and clk = '1';

    -- Volvemos a activar ini_c y a mitad, activamos sto_1
    ini_c <= '1';
    wait until clk'event and clk = '1';
    ini_c <= '0';
    wait until clk'event and clk = '1';

    wait for 500*T_CLK;
    wait until clk'event and clk = '1';
    sto_1 <= '1';
    wait until clk'event and clk = '1';
    sto_1 <= '0';
    wait until clk'event and clk = '1';
    wait for 700*T_CLK;


    -- Volvemos a activar ini_c y a mitad, activamos sto_2
    ini_c <= '1';
    wait until clk'event and clk = '1';
    ini_c <= '0';
    wait until clk'event and clk = '1';

    wait for 500*T_CLK;
    wait until clk'event and clk = '1';
    sto_2 <= '1';
    wait until clk'event and clk = '1';
```

```
   sto_2 <= '0';
   wait until clk'event and clk = '1';
   wait for 500*T_CLK;



   -- Fin del test
   wait until clk'event and clk = '1';
   end_sim <= true;
   wait;
 end process;
end test;
```