CNT_DAC

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity cnt_dac is
  port (
    CLK     : in  std_logic;
    RST     : in  std_logic;
    DATO1   : in  std_logic_vector(7 downto 0);
    DATO2   : in  std_logic_vector(7 downto 0);
    DATO_OK : in  std_logic;
    SYNC    : out std_logic;
    SCLK    : out std_logic;
    D1      : out std_logic;
    D2      : out std_logic);
end cnt_dac;

architecture RTL of cnt_dac is
  signal D1BD      : std_logic_vector(7 downto 0);
  signal D2BD      : std_logic_vector(7 downto 0);
  signal SCDATA    : std_logic_vector(3 downto 0);
  signal CEC       : std_logic;
  signal Q0        : std_logic;
  signal Q1        : std_logic;
  signal FinTX     : std_logic;
  signal Stado_Rep : std_logic;
  signal RE_CB     : std_logic;

  type MEF is (REP, TX, R1, R2);
  signal std_act, prox_std : MEF;

begin  -- RTL

  process (CLK, RST) is
  begin
    if RST = '1' then
      D1BD <= (others => '0');
    elsif CLK'event and CLK = '1' then
      if DATO_OK = '1' then
        D1BD(0) <= DATO1(0);
        D1BD(1) <= DATO1(1);
        D1BD(2) <= DATO1(2);
        D1BD(3) <= DATO1(3);
        D1BD(4) <= DATO1(4);
        D1BD(5) <= DATO1(5);
        D1BD(6) <= DATO1(6);
        D1BD(7) <= DATO1(7);
      end if;
    end if;
  end process;
```

```
process (CLK, RST) is
begin
  if RST = '1' then
    D2BD <= (others => '0');
  elsif CLK'event and CLK = '1' then
    if DATO_OK = '1' then
      D2BD(0) <= DATO2(0);
      D2BD(1) <= DATO2(1);
      D2BD(2) <= DATO2(2);
      D2BD(3) <= DATO2(3);
      D2BD(4) <= DATO2(4);
      D2BD(5) <= DATO2(5);
      D2BD(6) <= DATO2(6);
      D2BD(7) <= DATO2(7);
    end if;
  end if;
end process;

process (D1BD, SCDATA) is
begin
  case SCDATA is
                    when "0000" => D1 <= '0';
    when "0001" => D1 <= '0';
    when "0010" => D1 <= '0';
    when "0011" => D1 <= '0';
    when "0100" => D1 <= D1BD(0);
    when "0101" => D1 <= D1BD(1);
    when "0110" => D1 <= D1BD(2);
    when "0111" => D1 <= D1BD(3);
    when "1000" => D1 <= D1BD(4);
    when "1001" => D1 <= D1BD(5);
    when "1010" => D1 <= D1BD(6);
    when "1011" => D1 <= D1BD(7);
    when "1100" => D1 <= '0';
    when "1101" => D1 <= '0';
    when "1110" => D1 <= '0';
    when others => D1 <= '0';
  end case;
end process;

process (D2BD, SCDATA) is
begin
  case SCDATA is
                    when "0000" => D2 <= '0';
    when "0001" => D2 <= '0';
    when "0010" => D2 <= '0';
    when "0011" => D2 <= '0';
    when "0100" => D2 <= D2BD(0);
    when "0101" => D2 <= D2BD(1);
    when "0110" => D2 <= D2BD(2);
    when "0111" => D2 <= D2BD(3);
    when "1000" => D2 <= D2BD(4);
```

```
    when "1001" => D2 <= D2BD(5);
    when "1010" => D2 <= D2BD(6);
    when "1011" => D2 <= D2BD(7);
    when "1100" => D2 <= '0';
    when "1101" => D2 <= '0';
    when "1110" => D2 <= '0';
    when others => D2 <= '0';
  end case;
end process;

CEC <= '1' when Q0 = '1' and Q1= '0' else '0';

SCLK <= Q1;

process (CLK, RST, Stado_Rep) is
begin
          if RST = '1' then
            SCDATA <= (others => '0');        --pone a 0 pero seria 1 preguntar
  elsif Stado_Rep = '0' then
    SCDATA <= (others => '0');        --pone a 0 pero seria 1 preguntar
  elsif CLK'event and CLK = '1' then
    if CEC = '1' then
      if SCDATA = x"0" then
        SCDATA <= (others => '1');
      else
        SCDATA <= std_logic_vector(unsigned(SCDATA)-1);
      end if;
    end if;
  end if;
end process;

process (CLK, RST, RE_CB) is
  variable cnt : std_logic_vector(1 downto 0);
begin
  if RST = '1' then
    cnt := (others => '0');
    Q0  <= '0';
    Q1  <= '0';
  elsif RE_CB = '0' then
    cnt := (others => '0');
    Q0  <= '0';
    Q1  <= '0';
  elsif CLK'event and CLK = '1' then
    if cnt = "11" then
      cnt := (others => '0');
      Q0  <= cnt(0);
      Q1  <= cnt(1);
    else
      cnt := std_logic_vector(unsigned(cnt)+1);
      Q0  <= cnt(0);
      Q1  <= cnt(1);
    end if;
```

```
    end if;
  end process;

  process (DATO_OK, FinTX, std_act) is
  begin
    case std_act is
      when REP =>
        if DATO_OK = '1' then
          prox_std <= TX;
        else
          prox_std <= REP;
        end if;
      when Tx =>
        if FinTX = '1' then
          prox_std <= R1;
        else
          prox_std <= TX;
        end if;
      when R1 => prox_std <= R2;
      when R2 => prox_std <= REP;
    end case;
  end process;

  process (CLK, RST) is
  begin  -- process
    if RST = '1' then
      std_act <= REP;
    elsif CLK'event and CLK = '1' then
      std_act <= prox_std;
    end if;
  end process;

  process (std_act, DATO_OK) is
  begin  -- process
    case std_act is
      when REP => SYNC <= '1';
      when TX =>
        if DATO_OK = '0' then
          SYNC <= '0';
        else
          SYNC <= '1';
        end if;
      when R1 => SYNC <= '0';
      when R2 => SYNC <= '0';
    end case;
  end process;

  FinTX <= '1' when SCDATA = "0000" and std_act = TX and Q1 = '1' else '0';

  Stado_Rep <= '0' when std_act = REP else '1';
  RE_CB <= '0' when std_act = REP or DATO_OK = '1' else '1';
end RTL;
```