Sistemes Digitals
Control. 09 d'abril de 2015

Temps per a la resolució: 2 hores.

1 Protecció d'altes temperatures (70%)

Tenim una màquina funcionant tot el dia i l'hem de protegir d'un possible sobreescalfament. Per controlar-ho, tenim un sensor de temperatura que ens envia la temperatura en un bus de 8 bits síncronament amb un senyal de rellotge comú d'1kHz (dada vàlida en el flanc de pujada del rellotge). El valor decimal d'aquest bus és directament el valor en graus Celsius.
En el cas de que aquest valor arribi als 100 ºC hem d'aturar la màquina i alhora, també, hem d'activar un senyal d'alarma. Per tornar a posar la màquina en funcionament i desactivar l'alarma caldrà que una persona premi un botó de reset i que la temperatura hagi baixat dels 100 ºC.
Per tant, hem de dissenyar un sistema que tingui com a entrades els senyals de `clk`, `reset` i `temp` i com a sortida els senyals `stop` ('1' = màquina aturada) i `alarma`.
L'alarma serà un LED funcionant intermitentment. És a dir, el senyal que heu de crear per connectar al LED ha de ser un senyal periòdic de 10 Hz.
Per resoldre el problema respongueu les següents preguntes:

1. Dibuixeu un diagrama de blocs corresponent al *sistema complet*. Feu un disseny sintetitzable completament síncron;

2. Escriviu l'`entity` i l'`architecture` del sistema. Atenció: Definiu acuradament cadascun dels senyals interns del vostre disseny;

3. Dibuixeu un cronograma (incloent els senyals interns) que verifiqui el correcte funcionament del vostre sistema.

2 Qüestions 30%

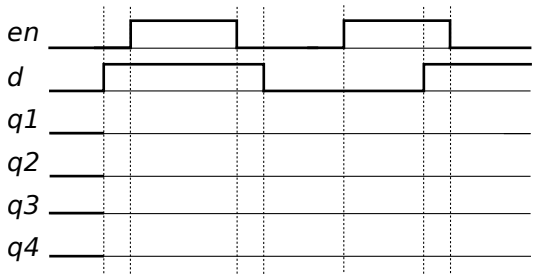
1. Volem crear dos comptadors. Un que compti fins a 19 cicles del rellotge d'entrada i un altre que compti fins a 9 però 19 vegades més lent que el primer.
   És correcte el codi següent tenint en compte el que volem aconseguir? Si és incorrecte corregiu-lo. (Pista: Penseu en si volem simular aquest codi.)

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

entity divisor is
port ( clk  : in std_logic;
       count1_out : out std_logic_vector(3 downto 0);
       count2_out : out std_logic_vector(3 downto 0));
end divisor;

architecture behav of divisor is
  signal count1, count2 : unsigned(3 downto 0) := "0000";
  signal tc : std_logic := '0';
begin
  process(clk)
  begin
    if rising_edge(clk) then
      if count1 = 18 then
        count1 <= (others => '0');
      else
        count1 <= count1 + 1;
      end if;
    end if;
  end process;
  tc <= '1' when count1 = 18;
  process(clk)
  begin
    if rising_edge(clk) then
      if tc = '1' then
        if count2 = 8 then
          count2 <= (others => '0');
        else
          count2 <= count2 + 1;
        end if;
       end if;
    end if;
  end process;
  count1_out <= std_logic_vector(count1);
  count2_out <= std_logic_vector(count2);
end behav;
```

2. En què consisteix el Pipelining?

3. Feu les següents operacions treballant amb vectors binaris amb signe i d'amplada de 6 bits. Indiqueu, també, si hi ha hagut Overflow i/o Carry de sortida:
   a) -5 + 30
   b) 15 + 17

4. Completeu el cronograma tenint en compte els diferents processos que es descriuen a continuació:




```
fig1: process(d,en)
begin
  if en='1' then
    q1<=d;
  end if;
end process fig1;

fig3: process(en)
begin
  if en='1' then
    q3<=d;
  end if;
end process fig3;
```

```
fig2: process(d)
begin
  if en='1' then
    q2<=d;
  end if;
end process fig2;

fig4: process(en)
begin
  q4<=d;
end process fig4;
```