INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 80 ns. En el cronograma se debe indicar para cada una de estas señales el instante de tiempo en que la señal cambia de valor así como su nuevo valor.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
  signal s1, s2, x3, x4, x5, x6 : std_logic;
begin
  s1 <= '1', '0' after 5 ns,'1' after 20 ns,
        '0' after 30 ns,'1' after 40 ns;
  s2 <= '0', '1' after 25 ns, '0' after 35 ns,
        '1' after 45 ns;
  Proc1: process
    variable valor : std_logic;
  begin
    wait for 5 ns;
    for i in 0 to 6 loop
      x3 <= s1 or s2;
      valor := s1 or s2;
      x4 <= valor;
      x5 <= x3;
      wait for 10 ns;
      end loop;
      wait;
  end process;
  x6 <= s1 or s2 after 12 ns;
end architecture crono1;
```

Pregunta 2 (3 puntos)

Escriba en VHDL la **architecture** que describe el comportamiento de un circuito contador binario de 4 bits que opera de modo síncrono en el flanco de subida de la señal de reloj (`clk`). El circuito tiene 5 señales de entrada de un bit y una señal de entrada de 4 bits (`d`). El circuito tiene una señal de salida de 4 bits (`q`). A continuación, se muestra la **entity** del circuito.

```
entity contador is
   port (  q : out std_logic_vector(3 downto 0);
           clk, reset : in  std_logic;
           sync_clr, en, load : in std_logic;
           d : in std_logic_vector(3 downto 0) );
end entity contador;
```

El orden de prioridad de las señales de entrada del circuito es: `reset`–`sync_clr`–`load`–`en`, siendo la señal `reset` la más prioritaria y la señal `en` la menos prioritaria. Las señales `reset`, `sync_clr`, `load` y `en` están activas a nivel alto, por lo que producen cambios en la salida del circuito sólo cuando tienen valor `'1'`. Por ejemplo, para que la señal `en` produzca un cambio en el circuito tiene que tener un valor `'1'` y las señales de entrada `reset`, `sync_clr` y `load` tienen que tener valor `'0'`.

Cuando la señal `reset` está activa, se produce un reseteo asíncrono del circuito y todos los bits de la señal de salida `q` se ponen a cero.

Cuando la señal `sync_clr` toma el valor `'1'` todos los bits de la señal de salida `q` se ponen a cero en el siguiente flanco de subida de la señal de reloj.

Cuando la señal `load` toma el valor `'1'`, se carga en la señal de salida `q` el valor de la señal de entrada `d`.

La señal `en` habilita la cuenta del circuito. Es decir, cuando tiene el valor `'1'` se incrementa el valor de la señal de salida `q` en uno cada ciclo de reloj. Sin embargo, cuando la señal `en` tiene el valor `'0'`, la señal de salida `q` no cambia de valor.

Pregunta 3 (2 puntos)

Escriba en VHDL un banco de pruebas que permita comprobar mediante inspección visual el comportamiento del circuito contador de la Pregunta 2. El banco de pruebas debe generar una señal de reloj de periodo 100 ns. Además, debe realizar consecutivamente los siguientes pasos:

- Primero, realiza un reseteo asíncrono del circuito entre los instantes 0 ns y 125 ns.
- Después, carga en la señal de salida el valor `"0010"`.
- Finalmente, realiza la cuenta hasta el valor `"0100"` y mantiene este valor.

Dibuja el cronograma de las señales de entrada y salida del circuito que se generará tras simular el banco de pruebas que ha escrito.

Pregunta 4 (3 puntos)

Escriba en VHDL la **architecture** que describe el comportamiento de un divisor de frecuencias por 6, con señal de reset asíncrona activa a nivel bajo.

Es decir, la señal de salida del circuito (`clk6`) tiene una frecuencia que es la sexta parte de la frecuencia de la señal de reloj de entrada (`clk`). Cuando la señal de `resetn` tiene valor `'0'`, la señal de salida `clk6` se pone a valor cero de forma asíncrona.

El código VHDL de la **entity** del divisor de frecuencias se muestra a continuación.

```
entity divisor_frecuencia_6 is port(
      clk6       :  out std_logic;
      clk, resetn:  in  std_logic );
end entity divisor_frecuencia_6;
```