INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
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 `x1`, `x2`, `x3`, `s`, `y` entre los instantes 0 y 50 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity crono is
end entity crono;

architecture crono of crono is
  signal x1, x3, s, y : std_logic;
  signal x2  : std_logic := '0';
begin
  x1 <= '1',
        '0' after 10 ns,
        '1' after 20 ns,
        '0' after 30 ns,
        '1' after 40 ns;
  x2 <= '0',
        '1' after 25 ns;
  x3 <= '0',
        '1' after 10 ns,
        '0' after 30 ns;
  Proc1: process (x1, x2, x3)
  begin
    s <= x1;
    y <= s or x3;
    s <= X2;
  end process;
end architecture crono;
```

Pregunta 2 (3 puntos)

Se pretende diseñar un circuito comparador cuya salida (`F`) vale '1' si el valor del número de cuatro bits de entrada (`x`) es menor que el número decimal 9. Los cuatro bits de entrada se interpretan como un número binario sin signo. La **entity** del circuito se muestra a continuación.

```
entity comparaXmenor9 is port
        ( F : out std_logic;
          x : in  std_logic_vector(3 downto 0) );
end entity comparaXmenor9;
```

Escriba en VHDL la **architecture** del circuito comparador de las dos formas siguientes:

2.a) (1 punto) Empleando únicamente asignaciones concurrentes y operadores lógicos.

2.b) (2 puntos) Describiendo su estructura. Dibuje el diagrama del circuito al nivel de puertas lógicas. Escriba en VHDL la **entity** y **architecture** de todos los tipos de puertas lógicas que contiene el diseño del comparador. Finalmente, escriba en VHDL la **architecture** del comparador usando las puertas lógicas que ha definido previamente.

Pregunta 3 (2 puntos)

Programe en VHDL el banco de pruebas del comparador que ha diseñado al resolver la Pregunta 2. El banco de pruebas debe comprobar el funcionamiento del circuito de forma exhaustiva. El banco de pruebas debe generar un conjunto de vectores de test, comprobar si la salida de la UUT es correcta, mostrar un mensaje cada vez que la salida de la UUT no sea correcta y mostrar un mensaje al finalizar el test en el que se indique el número total de salidas incorrectas.

Pregunta 4 (3 puntos)

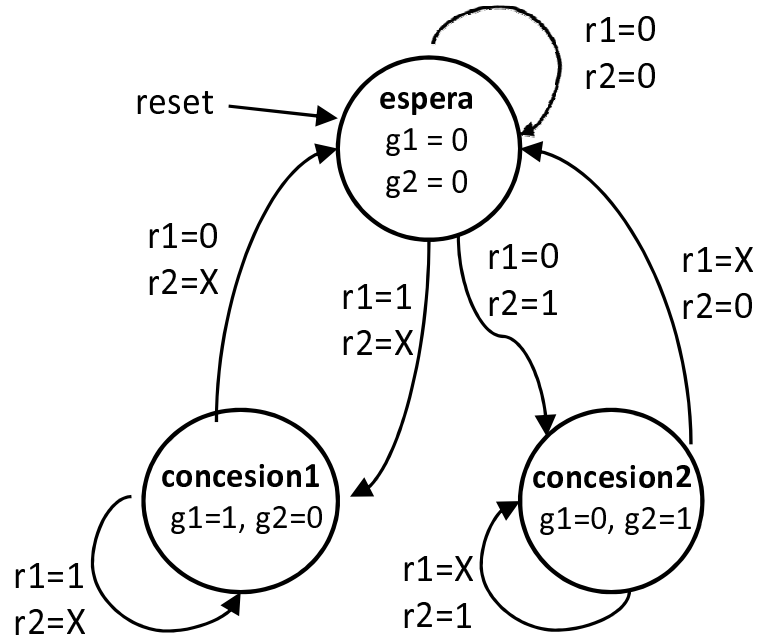
El circuito denominado *árbitro* permite controlar el acceso de varios dispositivos, denominados *clientes*, a un determinado recurso compartido. Este recurso compartido puede ser una memoria, un bus o cualquier componente de uso común. El objetivo del circuito árbitro es permitir en un momento dado el acceso de un único cliente al recurso. Para controlar el acceso de los clientes al recurso por medio del árbitro se usa un sistema de peticiones (*request*) y concesiones (*grant*).

Se quiere diseñar un árbitro que tenga dos clientes, que llamamos cliente1 y cliente2. La comunicación entre el cliente1 y el árbitro se realiza mediante las señales de petición (`r1`) y de concesión (`g1`). Análogamente, la comunicación entre el cliente2 y el árbitro se realiza mediante las señales `r2` y `g2`.

El cliente activa su señal de petición cuando quiere acceder al recurso compartido y el árbitro da acceso al cliente activando su señal de concesión correspondiente. Tras completar la tarea, el cliente libera el recurso y desactiva su señal de petición. Cuando ambos clientes realizan una petición simultánea, cliente1 tiene siempre la prioridad.

El comportamiento del árbitro puede describirse mediante una máquina de estados finitos con tres estados: espera, concesion1 y concesion2. El estado espera indica que el recurso está disponible y el árbitro está esperando peticiones. Los estados concesion1 y concesion2 indican que el recurso se ha dado al cliente1 y al cliente2, respectivamente. El árbitro ha de tener también una señal de reset asíncrona activa a nivel alto.

El diagrama de estados de esta máquina se muestra a continuación. Se ha empleado el símbolo X para indicar que el valor de la señal puede ser 0 ó 1.



Escriba en VHDL la **architecture** que describe el comportamiento de este árbitro de dos clientes como una máquina de estados de Moore síncrona, donde las transiciones tienen lugar en el flanco de subida de la señal de reloj.

La **entity** de este circuito árbitro se muestra a continuación.

```
entity arbitro is port
        ( g1, g2 : out std_logic;
          r1, r2, clk, reset : in  std_logic );
end entity arbitro;
```