

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES

Por favor, entregue esta primera hoja 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 clk, x1, x2, x3, x4, x5 y x6 entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
    signal x1, x2, x3, x4, x5, x6 : std_logic;
    signal clk : std_logic:='0';
begin
    x1 <= '0' , '1' after 10 ns, '0' after 20 ns, '1' after 25 ns;
    proc1: process (clk)
    begin
        if ( falling_edge(clk) ) then
            x4 <= x3;
            x3 <= x2;
            x2 <= x1;
        end if;
    end process;
    clk <= not clk after 10 ns;
    proc2: process (clk) is
        variable t1, t2, t3: std_logic;
    begin
        if (falling_edge(clk) ) then
            t1:= x1;
            t2:= t1;
            t3:= t2;
            x5 <= t1;
            x6 <= x5;
        end if;
    end process;
end architecture crono2;
```

Pregunta 2 (3 puntos)

Diseñe un contador binario síncrono de cuatro bits que opere en el flanco de subida de la señal de reloj. El contador tiene la siguiente **entity**:

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

La señal de salida *q* indica el valor actual de la cuenta. La señal de entrada *d* se carga en paralelo cuando se habilita la carga. La señal de entrada *reset* es activa a nivel alto y realiza un reset asíncrono del circuito poniendo la cuenta a cero. El contador tiene unas señales de entrada que permiten, en el flanco de subida de la señal de reloj, poner la cuenta a cero (reset síncrono), cargar un valor específico en la cuenta (señal de entrada *d*), activar la cuenta o pausarla, tal como se indica en la siguiente tabla. Cuando la cuenta está activada, se pasa cíclicamente por los valores “0000”, “0001”, ..., “1110”, “1111”.

<i>syn_clr</i>	<i>load</i>	<i>en</i>	Operación
1	-	-	Pone el contador a cero (reset síncrono)
0	1	-	Carga en paralelo
0	0	1	Activa la cuenta
0	0	0	Pausa la cuenta

En el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

```
IEEE.std_logic_1164
IEEE.numeric_std
```

Pregunta 3 (2 puntos)

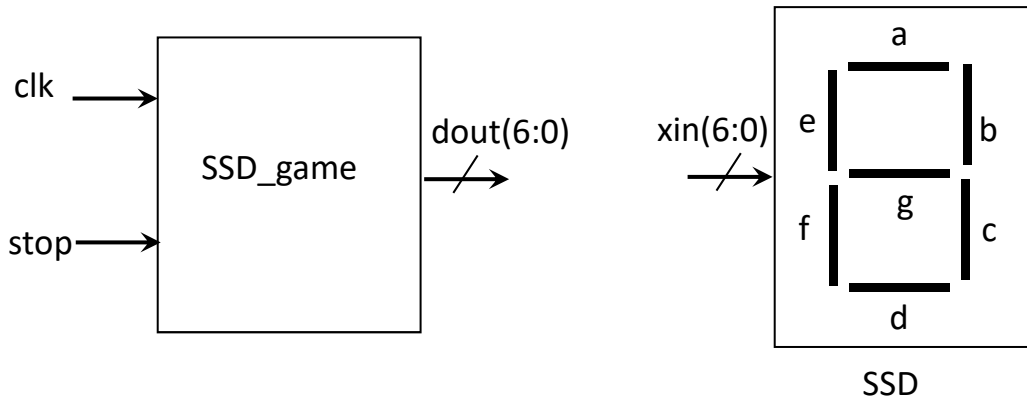
Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`clk`) debe tener un periodo de 20 ns e inicialmente valer '0'. El primer flanco de subida de la señal de reloj se ha de producir en el instante 10 ns. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset.* La señal de reset ha de tener el valor '1' durante los primeros 15 ns.
2. *Mantener la cuenta activa durante tres periodos de la señal de reloj.*
3. *Cargar en la cuenta el valor "1001".*
4. *Pausar la cuenta.*

El programa de test debe mostrar mensajes de error en el caso de que la señal de salida no tome el valor esperado. Dibuje el cronograma de las señales aplicadas al circuito y las salidas esperadas.

Pregunta 4 (3 puntos)

Se quiere diseñar un circuito denominado `SSD_game` para poder conectarse con un display de siete segmentos (SSD). En la siguiente figura se muestra el circuito a diseñar y el SSD con el que se pretende conectar.



El circuito `SSD_game` es un circuito síncrono que opera en el flanco de subida de la señal de reloj. El circuito contiene dos señales de entrada: la señal de reloj `clk` y la señal asíncrona `stop`. El circuito tiene una señal de salida de 7 bits llamada `dout` que se puede conectar con el circuito SSD. Consideramos que la señal de reloj que se va a conectar a la entrada del circuito tiene una frecuencia de 1 kHz (periodo de 1 ms).

El objetivo de los bits 0 a 6 de la señal `dout` es iluminar (valor '1') o apagar (valor '0') los segmentos `a`, `b`, `c`, `d`, `e`, `f` y `g` del circuito SSD, respectivamente. Por ejemplo, cuando `dout(6) = '1'` el segmento `g` está iluminado y cuando `dout(6) = '0'` el segmento `g` está apagado.

Siempre que la señal `stop` tenga el valor '0', el circuito genera unos valores de la señal `dout` que hace que se enciendan cíclicamente con cada flanco de subida de la señal de reloj los siguientes segmentos del SSD (mientras el resto de segmentos se mantienen apagados): segmento `a`, segmentos `b` y `c`, segmento `d`, segmentos `d` y `e`, segmento `e`, segmentos `e` y `f`.

Si la señal `stop` tiene el valor '1', el circuito pasa asíncronamente a un estado en que sólo esté encendido el segmento `a` del display. El circuito permanece en ese estado hasta que la señal `stop` tenga el valor '0'.

El sistema tiene que permanecer en los estados donde sólo hay un segmento del display iluminado durante un tiempo de 80 ms, y en los estados donde hay dos segmentos del display iluminados durante 30 ms.

Programa en VHDL el circuito `SSD_game` describiendo su comportamiento como una máquina de estados de tipo Moore. Dibuje el diagrama de estado correspondiente al diseño realizado.