Asignaturas:
Electrónica Digital (GITI)

Fecha: 18/12/2017
Examen: PEC diciembre 2017

CUESTIÓN 1 (5 puntos)

Se desea implementar un velocímetro para una motocicleta, empleando para ello un *sensor* (que generará un pulso de duración 1 ciclo de reloj cada vez que la rueda delantera de la motocicleta realice una vuelta completa) y una FPGA. En la FPGA se implementará un circuito que recibiendo los pulsos del *sensor* activará una tira de 8 LEDs, de manera proporcional a la velocidad de la motocicleta. La frecuencia de la señal de reloj que recibe la FPGA son 50 MHz. Se pide:

a) Describir un módulo VHDL que a partir de la salida del sensor, el reloj de la FPGA y una señal de reset, genere una señal de 6 bits con el número de vueltas (codificado en binario) que ha dado la rueda de la motocicleta en el último segundo. Esta salida deberá actualizarse al final de cada segundo, manteniéndose estable durante todo el segundo siguiente. El máximo número de vueltas que se considerará es 63.

b) Usando la descripción del apartado a como un componente jerárquico, describa la lógica que controla la activación de la tira de LEDs, de tal manera que el primer LED se activará cuando la velocidad proporcionada por el bloque anterior esté por debajo de 8 vueltas por segundo, el segundo se activará cuando la entrada entré entre 8 y 15 vueltas, y así sucesivamente hasta los 8 LEDs que componen la tira.

CUESTIÓN 2 (2 puntos)

Describir en VHDL (entidad y arquitectura) la siguiente máquina de estados:

CUESTIÓN 3 (3 puntos)

Describir un módulo VHDL que se corresponda con un circuito que genere la secuencia síncrona **11011001** de forma cíclica, de estas tres maneras posibles:

c) Usando como base un contador binario síncrono

d) Usando como base un contador Johnson

e) Usando como base un contador en anillo

***Duración del examen*: 1 hora y 30 minutos.**

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity velocimetro is
    Port ( clk          : in STD_LOGIC;
           reset        : in STD_LOGIC;
           sensor       : in STD_LOGIC;
           VueltasOut   : out std_logic_vector(5 downto 0));
end velocimetro;

architecture Behavioral of velocimetro is

    constant Conta1segMAX : integer := 50*(10**6) - 1;
    signal conta1seg : integer range 0 to Conta1segMAX;
    signal UnSeg : std_logic;

    signal Vueltas: unsigned(5 downto 0);

begin

    -- Divisor de Frecuencia a 1 segundo
    DivFreq: process(clk,reset)
    begin
        if reset = '1' then
            conta1seg <= 0;
        elsif clk'event and clk = '1' then
            if conta1seg < Conta1segMAX then
                conta1seg <= conta1seg + 1;
            else
                conta1seg <= 0;
            end if;
        end if;
    end process;
    UnSeg <= '1' when conta1seg = Conta1segMAX else '0';

    -- Proceso en el que se cuentan las vueltas y se actualiza la salida (1 vez por segundo)
    ContaVueltas: process(clk,reset)
    begin
        if reset = '1' then
            Vueltas     <= "000000";
            VueltasOut  <= "000000";
        elsif clk'event and clk = '1' then
            if Unseg = '1' then
                VueltasOut <= std_logic_vector(Vueltas);
                Vueltas <= "000000";
            elsif sensor = '1' then
                    Vueltas <= Vueltas + 1;
            end if;
        end if;
    end process;

end Behavioral;
```

```
 1  library IEEE;
 2  use IEEE.STD_LOGIC_1164.ALL;
 3  use IEEE.NUMERIC_STD.ALL;
 4
 5  entity TopVelocimetro is
 6      Port ( clk : in STD_LOGIC;
 7             reset : in STD_LOGIC;
 8             sensor : in STD_LOGIC;
 9             leds : out STD_LOGIC_VECTOR(7 DOWNTO 0)
10           );
11  end TopVelocimetro;
12
13  architecture Behavioral of TopVelocimetro is
14
15      component velocimetro is
16          Port ( clk : in STD_LOGIC;
17                 reset : in STD_LOGIC;
18                 sensor : in STD_LOGIC;
19                 VueltasOut   : out std_logic_vector(5 downto 0));
20      end component;
21
22      signal VueltasOut : std_logic_vector(5 downto 0);
23      signal uV : unsigned(5 downto 0);
24
25  begin
26
27      -- Instancia del componente velocímetro
28      instVelo: velocimetro
29                port map(
30                 clk => clk,
31                 reset => reset,
32                 sensor => sensor,
33                 VueltasOut => VueltasOut
34                 );
35
36      --Transformación a unsigned para facilitar la codificación
37      uV <= unsigned(VueltasOut);
38
39      leds <= "00000001" when uV < 8 else
40              "00000010" when uV < 16 else
41              "00000100" when uV < 24 else
42              "00001000" when uV < 32 else
43              "00010000" when uV < 40 else
44              "00100000" when uV < 48 else
45              "01000000" when uV < 56 else
46              "10000000";
47  end Behavioral;
48
```

```
------------------------
-- Electrónica Digital --
-- PEC diciembre 2017  --
-- Cuestión 3          --
------------------------

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

entity c3 is
    port (
        clk   : in  std_logic;
        reset : in  std_logic;
        seq   : out std_logic
    );
end c3;

-----------------------------------------
-- La solución comienza a partir de aquí --
-----------------------------------------

-- a) Base: contador binario síncrono
architecture cnt_binario of c3 is
    signal cnt : unsigned(2 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            cnt <= (others => '0');
        elsif clk'event and clk = '1' then
            cnt <= cnt + 1;
        end if;
    end process;

    with cnt select
        seq <= '1' when "000",
               '1' when "001",
               '0' when "010",
               '1' when "011",
               '1' when "100",
               '0' when "101",
               '0' when "110",
               '1' when "111",
               '-' when others;

end cnt_binario;

-- b) Base: contador Johnson
architecture cnt_johnson of c3 is
    signal shiftreg : std_logic_vector(3 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            shiftreg <= (others => '0');
        elsif clk'event and clk = '1' then
            shiftreg <= shiftreg(2 downto 0) & not shiftreg(3);
        end if;
    end process;

    with shiftreg select
        seq <= '1' when "0000",
               '1' when "0001",
               '0' when "0011",
               '1' when "0111",
               '1' when "1111",
               '0' when "1110",
               '0' when "1100",
               '1' when "1000",
               '-' when others;
```

```vhdl

end cnt_johnson;

-- c) Base: contador en anillo
architecture cnt_anillo of c3 is
    signal shiftreg : std_logic_vector(7 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            shiftreg <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            shiftreg <= shiftreg(6 downto 0) & shiftreg(7);
        end if;
    end process;

    with shiftreg select
        seq <= '1' when "00000001",
               '1' when "00000010",
               '0' when "00000100",
               '1' when "00001000",
               '1' when "00010000",
               '0' when "00100000",
               '0' when "01000000",
               '1' when "10000000",
               '-' when others;

end cnt_anillo;

```