INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Convocatoria extraordinaria 2023

Ejercicio 1

Dada las dos funciones lógicas (`F` y `G`) mostradas a continuación, que dependen de 3 variables (`x`, `y` y `z`):

```
F = (x and y) or z
G = x or z
```

1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito.

1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

1.c) (0.5 puntos) Dibuje el diagrama de un circuito que implemente estas dos funciones lógicas al nivel de puertas lógicas. A continuación, escriba en VHDL la **entity** y la **architecture** de cada una de las puertas lógicas que componen el circuito que acaba de dibujar.

1.d) (1 punto) Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

1.e) (1 punto) Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, las salidas de los circuitos diseñados en los Apartados 1.b y 1.d. Compruebe mediante inspección visual que los dos diseños funcionan correctamente. Incluya en la memoria los dos

cronogramas obtenidos al realizar la simulación del banco de pruebas con los circuitos diseñados en los Apartados 1.b y 1.d.

Solución al Ejercicio 1

El Código VHDL 1.1 y 1.2 es una posible solución a los Apartados 1.a y 1.b.

El diagrama al nivel de puertas lógicas pedido en el Apartado 1.c está representado en la Figura 1.1.

El Código VHDL 1.3 – 1.4 es una posible descripción de las puertas lógicas AND y OR de dos entradas. El Código VHDL 1.5 es una posible descripción estructural del circuito, que corresponde con el diagrama mostrado en la Figura 1.1.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity funcLog_F_G is
5    port ( F, G       : out std_logic;
6           x, y, z : in  std_logic );
7 end entity funcLog_F_G;
```

Código VHDL 1.1: Apartado 1.a: **entity** del circuito.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 architecture funcLog_F_G_Comp of funcLog_F_G is
5 begin
6    F <= (x and y) or (z);
7    G <= x or z;
8 end architecture funcLog_F_G_Comp;
```

Código VHDL 1.2: Apartado 1.b: **architecture** que describe el comportamiento.

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

entity and2 is
   port ( y0     : out std_logic;
          x0, x1 : in  std_logic );
end entity and2;

architecture and2 of and2 is
begin
   y0 <= x0 and x1;
end architecture and2;
```

Código VHDL 1.3: Apartado 1.c: puerta AND de dos entradas.

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

entity or2 is
  port ( y0     : out std_logic;
         x0, x1 : in  std_logic );
end entity or2;

architecture or2 of or2 is
begin
   y0 <= x0 or x1;
end architecture or2;
```

Código VHDL 1.4: Apartado 1.c: puerta OR de dos entradas.

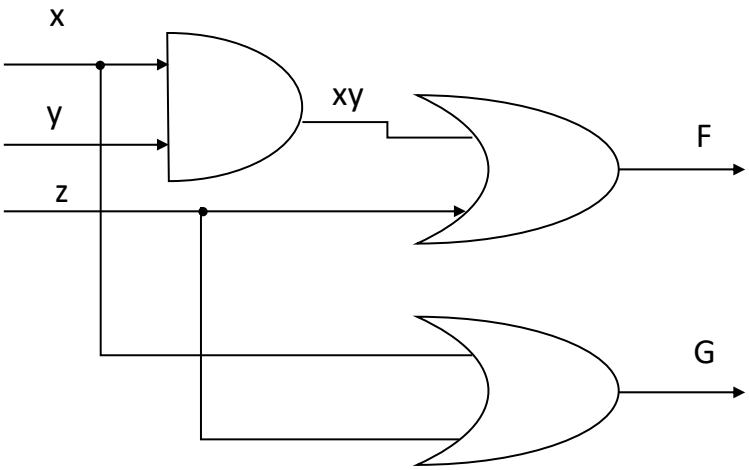

Figura 1.1: Apartado 1.c: diagrama al nivel de puertas lógicas.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  architecture funcLog_F_G_Estruc of funcLog_F_G is
5     signal xy : std_logic;
6
7     -- Declaración de las clases de los componentes
8     component and2 is
9        port ( y0     : out std_logic;
10              x0, x1 : in std_logic );
11    end component and2;
12
13
14    component or2 is
15       port ( y0     : out std_logic;
16              x0, x1 : in std_logic );
17    end component or2;
18
19 begin
20    -- Instanciación y conexión de los componentes
21    g0 : component and2 port map (xy, x, y);
22    g1 : component or2 port map (F, xy, z);
23    g2 : component or2  port map (G, x, z);
24 end architecture funcLog_F_G_Estruc;
```

Código VHDL 1.5: Apartado 1.d: descripción estructural al nivel de puertas lógicas.

Finalmente, el Código VHDL 1.6 es un banco de pruebas para los diseños del circuito que implementa las dos funciones lógicas. En las Figuras 1.2–1.3 se muestran las formas de onda obtenidas al simular el banco de pruebas con los dos diseños anteriormente realizados. La comprobación de que el diseño funciona correctamente debe hacerse en este caso mediante inspección visual.

```vhdl
-- Banco de pruebas
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_funcLog_F_G is
end entity bp_funcLog_F_G;

architecture bp_funcLog_F_G of bp_funcLog_F_G is
   signal y0, y1         : std_logic;  -- Conectar salidas UUT
   signal x, y, z : std_logic;  -- Conectar entradas UUT

   component funcLog_F_G is
      port ( F, G       : out std_logic;
             x, y, z : in std_logic);
   end component funcLog_F_G;

begin
   -- Instanciar y conectar UUT
   uut : component funcLog_F_G port map
         ( F => y0, G => y1,
           x => x, y => y, z => z);

   gen_vec_test : process
      variable test_in : unsigned (2 downto 0); -- Vector de test
   begin
      test_in := B"000";
      for count in 0 to 7 loop
         z <= test_in(2);
         y <= test_in(1);
         x <= test_in(0);
         wait for 10 ns;
         test_in := test_in + 1;
      end loop;
      wait;
   end process gen_vec_test;
end architecture bp_funcLog_F_G;
```

Código VHDL 1.6: Apartado 1.e: banco de pruebas.

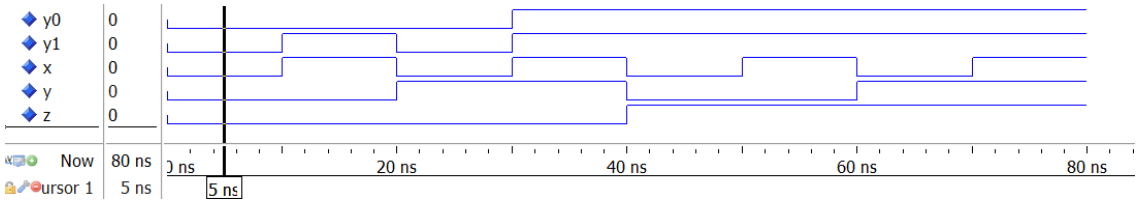

Figura 1.2: Apartado 1.e: simulación del banco de pruebas del diseño describiendo el comportamiento del circuito.

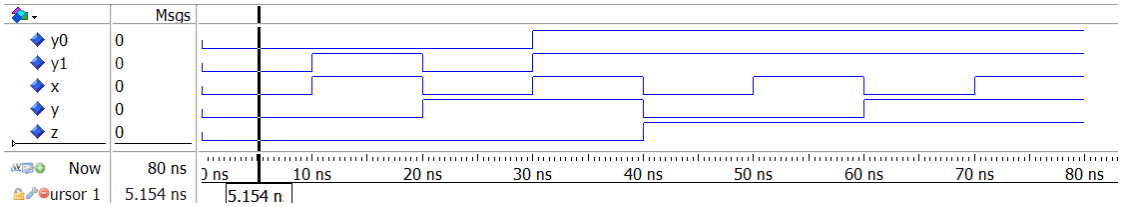

Figura 1.3: Apartado 1.e: simulación del banco de pruebas del diseño describiendo la estructura del circuito.

Ejercicio 2

Un contador decimal progresivo de tres dígitos realiza la cuenta de 000 a 999 de forma cíclica, pasando de un número al consecutivo en el flanco de subida de la señal de reloj siempre que la señal de cuenta esté habilitada y la señal de reset esté a nivel bajo. Como el contador es cíclico, se considera que el número que sigue al 999 es el 000. El circuito tiene un reset asíncrono activo a nivel alto. Cuando el circuito se resetea, la cuenta toma el valor 000.

Cada dígito decimal se representa por un número binario de 4 bits. Por ejemplo, el número decimal 139 es "0001 0011 1001". El número siguiente al 139 en la secuencia contadora es el 140, que se representa como "0001 0100 0000".

La **entity** del circuito contador se muestra a continuación.

```
entity contador_decimal is
   port ( d1, d10, d100  : out std_logic_vector(3 downto 0);
          clk   : in  std_logic;
          en    : in std_logic;
          reset  : in  std_logic );
end contador_decimal;
```

Las señales de salida tienen como valor el resultado de la cuenta. La señal `d1` se corresponde con el dígito menos significativo de la cuenta y la señal `d100` se corresponde con el dígito más significativo. Por ejemplo, los valores de las señales `d1`, `d10` y `d100` al representar al número 140 son `d100 = "0001"`, `d10 = "0100"` y `d1 = "0000"`. La señal `en` permite habilitar y deshabilitar la cuenta. La cuenta está habilitada cuando el valor de la señal `en` es '1' y está desabilitada cuando su valor es '0'. La señal `reset` es la señal de reset del circuito.

2.a) (3 puntos) Diseñe en VHDL la **architecture** que describe el comportamiento del circuito secuencial contador.

2.b) (3 puntos) Programe en VHDL el banco de pruebas del circuito contador que ha diseñado en el anterior apartado. El banco de pruebas debe generar una señal de reloj de periodo 0.01 s que sea la entrada al circuito contador. El banco de pruebas ha de realizar consecutivamente las siguientes pruebas:

- Resetea del circuito comprobando que la salida del circuito es la correcta.
- Pone la señal `en` a '1' y comprueba que la salida del circuito pasa consecutivamente por los valores 0 a 999 y de 999 a 0.
- Pone la señal `en` a '0' y comprueba que la salida del circuito no cambia de valor una vez pasado al menos un ciclo de reloj.

El banco de pruebas debe comprobar que las salidas de la UUT coinciden con las salidas esperadas, mostrando el correspondiente mensaje de error en caso contrario. Emplee este banco de pruebas para comprobar el diseño realizado al contestar el Apartado 2.a. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas del circuito diseñado en el Apartado 2.a entre los instantes 0 s y 2 s.

Solución al Ejercicio 2

El Código VHDL 1.7 muestra el diseño del circuito contador.

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

architecture contador_decimal of contador_decimal is
  signal d1_reg, d10_reg, d100_reg: unsigned(3 downto 0);
begin
  process(clk, reset)
    begin
    if (reset = '1') then
      d1_reg <= (others=>'0');
      d10_reg <= (others=> '0');
      d100_reg <= (others => '0');
    elsif (en = '1' and rising_edge(clk)) then
      if d1_reg /= "1001" then
          d1_reg <= d1_reg+1;
      else -- alcanza el 9
          d1_reg <= "0000";
          if (d10_reg /= "1001") then
             d10_reg <= d10_reg+1;
          else --alcanza 99
             d10_reg <= "0000";
             if d100_reg/="1001" then
                d100_reg <= d100_reg+1;
             else
                d100_reg <= "0000";
             end if; --d100_reg
          end if;--d10_reg
      end if; --d1_reg
    end if; -- else
  end process;
  d1 <= std_logic_vector(d1_reg);
  d10 <= std_logic_vector(d10_reg);
  d100 <= std_logic_vector(d100_reg);
end contador_decimal;
```

Código VHDL 1.7: Apartado 2.a: diseño del circuito contador.

El banco de pruebas del circuito combinacional se muestra en el Código VHDL 1.8. El cronograma obtenido al simular el banco de pruebas, usando como circuito a testear el diseño del Apartado 2.a, se muestra en la Figura 1.4.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_contador is
end entity bp_contador;
architecture bp_contador of bp_contador is
  constant PERIODO   : time       := 0.01 sec; -- Reloj
  signal   d1, d10, d100 : std_logic_vector (3 downto 0);
  signal   clk       : std_logic  := '0';    -- Entradas UUT
  signal   en    : std_logic;
  signal   reset : std_logic;

  component contador_decimal is
    port( d1, d10, d100  : out std_logic_vector(3 downto 0);
            clk  : in  std_logic;
            en   : in std_logic;
            reset  : in  std_logic    );
  end component contador_decimal;
  -- Procedimiento para comprobar las salidas
   procedure comprueba_salidas
      ( esperado_z             :       std_logic_vector(3 downto 0);
        actual_z               :       std_logic_vector(3 downto 0)) is
   begin
       if ( esperado_z /= actual_z ) then
          report "ERROR: Salida esperada ("        &
                 std_logic'image(esperado_z(3))    &
                 std_logic'image(esperado_z(2))    &
                 std_logic'image(esperado_z(1))    &
                 std_logic'image(esperado_z(0))    &
                 "), salida actual ("              &
                 std_logic'image(actual_z(3))      &
                 std_logic'image(actual_z(2))      &
                 std_logic'image(actual_z(1))      &
                 std_logic'image(actual_z(0))      &
                 "), instante: "                   &
                 time'image(now);
       end if;
    end procedure comprueba_salidas;
begin
  -- Instanciar y conectar UUT
  uut : component contador_decimal port map
                  (d1, d10, d100, clk, en, reset);

  clk <= not clk after (PERIODO/2);
  reset <= '1', '0' after periodo/4;
    gen_vec_test : process
       variable vd1,vd10,vd100 : integer;
       variable esperado_d1, esperado_d10, esperado_d100:
   std_logic_vector(3 downto 0);
```

```
49      begin
50          en <= '1';
51          wait for periodo/4;
52          for count in 0 to 1000 loop
53              if count <1000 then
54                vd100 := count/100;
55                vd10 := (count-vd100*100);
56                vd10 := vd10/10;
57                vd1 := (count-100*vd100-vd10*10);
58              else
59                vd1 := 0;
60                vd10 := 0;
61                vd100 := 0;
62                en <= '0';
63              end if;
64              esperado_d1 := std_logic_vector(to_unsigned(vd1,4));
65              esperado_d10 := std_logic_vector(to_unsigned(vd10,4));
66              esperado_d100 := std_logic_vector(to_unsigned(vd100,4));
67              comprueba_salidas(esperado_d1, d1);
68              comprueba_salidas(esperado_d10, d10);
69              comprueba_salidas(esperado_d100, d100);
70              wait for periodo;
71          end loop;
72          wait for 2*periodo;
73          comprueba_salidas(esperado_d1, d1);
74          comprueba_salidas(esperado_d10, d10);
75          comprueba_salidas(esperado_d100, d100);
76          wait;
77      end process gen_vec_test;
78  end architecture bp_contador;
```

Código VHDL 1.8: Apartado 2.b: banco de pruebas del circuito.

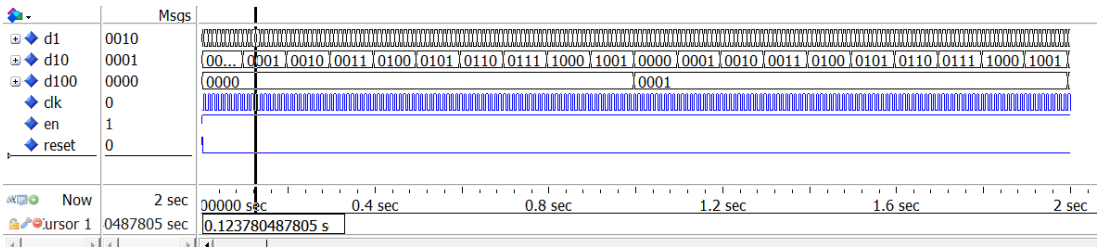

Figura 1.4: Cronograma del Apartado 2.b comprobando el comportamiento del circuito contador.