INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2014

EJERCICIO 1

En la Figura 1.1 se muestra el símbolo lógico de un circuito digital cuya función es contabilizar el número de señales de entrada que tienen valor par. Tanto las señales de entrada como de salida del circuito se interpretan como números binarios sin signo. Si una señal de entrada es "00" ó "10", es par. Si una señal de entrada es "01" ó "11", es impar. La salida de este circuito indica el número de señales pares existentes en la entrada (0, 1 ó 2). Si las dos entradas son "01" y "11", entonces la salida del circuito vale 0 ("00"). Si las señales de entrada son "00" y "01", entonces la salida del circuito vale 1 ("01"). Si las dos entradas son "00" y "10", entonces la salida del circuito vale 2 ("10").

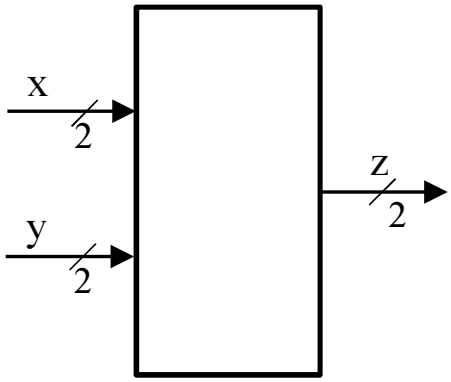


Figura 1.1: Entradas y salida del circuito del Ejercicio 1.

1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito digital empleando el mismo nombre para las señales que el mostrado en la Figura 1.1.

1.b) (1 punto) Escriba la tabla de la verdad del circuito digital. A partir de dicha tabla de la verdad, obtenga la función lógica que describe la salida (z) en

función de las entradas (x e y). A continuación, escriba en VHDL una **architecture** que describa el *comportamiento* de un circuito que implemente dicha función lógica.

1.c) (0.5 puntos) Dibuje el diagrama al nivel de puertas lógicas de un circuito que implemente esta función. A continuación, escriba en VHDL la **entity** y la **architecture** de cada una de las puertas lógicas que componen el diagrama 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, la salida de los circuitos diseñados en los Apartados 1.b y 1.d. Compruebe mediante inspección visual que los dos diseños funcionan correctamente.

Solución al Ejercicio 1

La **entity** del circuito solución al Ejercicio 1 se muestra en Código VHDL 1.1.

```
-----------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
entity contadorPar is
   port ( z   : out std_logic_vector (1 downto 0);
          x, y : in std_logic_vector (1 downto 0));
end entity contadorPar;
-----------------------------------
```

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

La tabla de verdad del circuito es la siguiente:

x(1)	x(0)	y(1)	y(0)	z(1)	z(0)
0	0	0	0	1	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	0
1	1	1	0	0	1
1	1	1	1	0	0

El Código VHDL 1.2 muestra la **architecture** del circuito describiendo su comportamiento.

La Figura 1.2 muestra el diagrama del circuito empleando una puerta AND, una puerta XOR y dos inversores.

El código VHDL de la **entity** y la **architecture** de las tres puertas lógicas empleadas se muestra en Código VHDL 1.3, 1.4 y 1.5, respectivamente.

El Código VHDL 1.6 muestra la **architecture** del circuito describiendo estructura.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture contadorPar of contadorPar is
begin
    z(1) <= (not x(0)) and (not y(0));
    z(0) <= x(0) xor y(0);
end architecture contadorPar;
-------------------------------------
```

Código VHDL 1.2: Solución al Apartado 1.b: **architecture** del circuito describiendo su comportamiento.

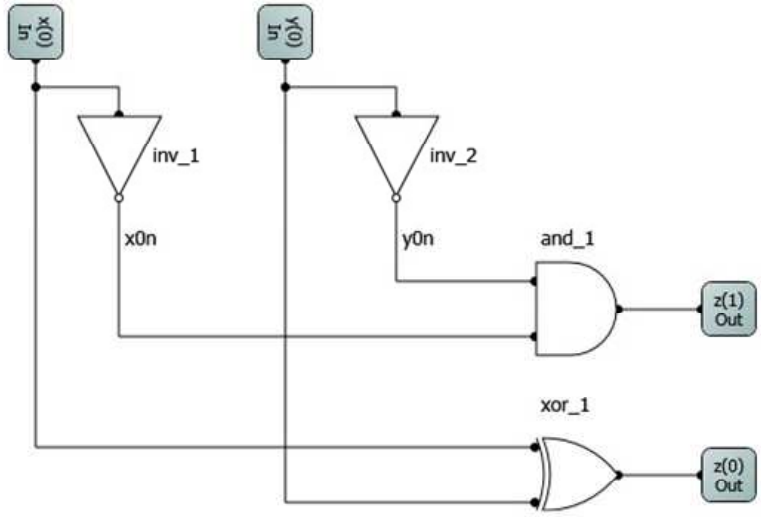


Figura 1.2: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

```
-------------------------------------
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: Puerta AND lógica.

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.7.

```
-------------------------------------
-- OR exclusiva de 2 entradas: xor2
library IEEE; use IEEE.std_logic_1164.all;

entity xor2 is port
   ( y0       : out std_logic;
     x0, x1 : in    std_logic );
end entity xor2;

architecture xor2 of xor2 is
begin
   y0 <= x0 xor x1;
end architecture xor2;
-------------------------------------
```

Código VHDL 1.4: Puerta XOR lógica.

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

entity not1 is
     port ( y0   : out std_logic;
            x0   : in    std_logic );
end entity not1;

architecture not1 of not1 is
begin
   y0 <= not x0;
end architecture not1;
-------------------------------------
```

Código VHDL 1.5: Puerta NOT lógica.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture contadorPar_Estruc of contadorPar is
   signal x0n, y0n: std_logic;
-- Declaración de las clases de los componentes
   component and2 is
      port (   y0 : out std_logic ;
      x0, x1 : in std_logic);
   end component and2;
   component not1 is
      port ( y0 : out std_logic;
                x0 : in std_logic );
   end component not1;
   component xor2 is
      port ( y0 : out std_logic;
          x0, x1 : in std_logic );
   end component xor2;
begin
-- Instanciación y conexión de los componentes
   inv_1 : component not1 port map (x0n, x(0));
   inv_2 : component not1 port map (y0n, y(0));
   xor_1 : component xor2 port map (z(0), x(0), y(0));
   and_1 : component and2 port map (z(1), x0n, y0n);
end architecture contadorPar_Estruc;
-----------------------------------
```

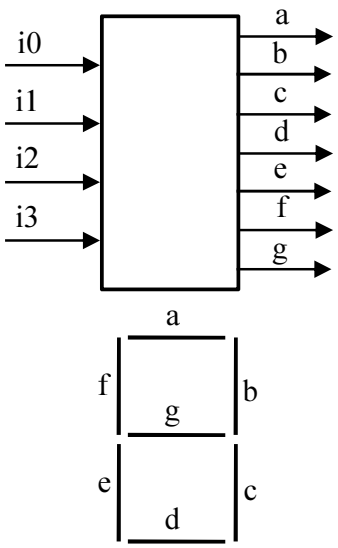
Código VHDL 1.6: Solución al Apartado 1.d: **architecture** del circuito describiendo su estructura.

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

entity bp_contadorPar is
end entity bp_contadorPar;

architecture bp_contadorPar of bp_contadorPar is
   signal z : std_logic_vector (1 downto 0); -- Conectar salidas UUT
   signal x, y: std_logic_vector(1 downto 0); -- Conectar entradas UUT

   component contadorPar is port
    ( z : out std_logic_vector (1 downto 0);
      x, y : in std_logic_vector(1 downto 0));
   end component contadorPar;

begin
-- Instanciar y conectar UUT
   uut : component contadorPar port map
   ( z=> z, x => x, y => y);

   gen_vec_test : process
     variable test_in : unsigned (3 downto 0); -- Vector de test
   begin
     test_in := B"0000";
     for count in 0 to 15 loop
       y(1) <= test_in(3);
       y(0) <= test_in(2);
       x(1) <= test_in(1);
       x(0) <= test_in(0);
       wait for 10 ns;
       test_in := test_in + 1;
     end loop;
     wait;
   end process gen_vec_test;
end architecture bp_contadorPar;
-----------------------------------
```

Código VHDL 1.7: Solución al Apartado 1.e: banco de pruebas del circuito.

EJERCICIO 2

En la Figura 1.3 se muestra el símbolo lógico de un circuito combinacional conversor de código BCD a 7 segmentos, el display de 7 segmentos y la tabla de operaciones del circuito.



i3	i2	i1	i0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1

Figura 1.3: Símbolo lógico del conversor código BCD a 7 segmentos (en la parte superior izquierda de la figura), tabla de operaciones (en la parte derecha) y display de siete segmentos (en la parte inferior izquierda).

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

```
entity BCDto7seg is
   port ( a, b, c, d, e, f, g  : out std_logic;
          i3, i2, i1, i0  : in  std_logic );
end entity BCDto7seg;
```

2.a) (1 punto) Dibuje el diagrama al nivel de puertas lógicas del circuito conversor. Puede usar todos los tipos de puertas lógicas que estime convenientes. A continuación, escriba en VHDL la **entity** y la **architecture** de cada una de las puertas lógicas que componen el diagrama que acaba de dibujar.

2.b) (2.5 puntos) Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

2.c) (2.5 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito conversor. El banco de pruebas debe comparar las salidas de la UUT con las salidas esperadas, mostrando el correspondiente mensaje de error en caso de que las salidas obtenidas de la UUT no correspondan con las esperadas. Emplee este banco de pruebas para comprobar el diseño realizado al contestar el Apartado 2.b.

Solución al Ejercicio 2

La Figura 1.4 muestra el diagrama del circuito empleando inversores, puertas AND de dos y tres entradas, puertas XNOR de dos entradas, puertas XOR de dos entradas y puertas OR de dos, tres y cuatro entradas.

El código VHDL de la **entity** y la **architecture** de las puertas lógicas empleadas se muestra en Código VHDL 1.5, 1.3, 1.8, 1.9, 1.4, 1.10, 1.11 y 1.12, respectivamente.

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

entity and3 is
   port ( y0 : out std_logic;
          x0, x1, x2 : in std_logic );
end entity and3;

architecture and3 of and3 is
begin
   y0 <= x0 and x1 and x2;
end architecture and3;
-------------------------------------
```

Código VHDL 1.8: Puerta AND lógica de tres entradas.

La **architecture** describiendo la estructura del circuito se muestra en Código VHDL 1.13–1.14.

Finalmente, el banco de pruebas del circuito conversor se muestra en Código VHDL 1.15–1.16.

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

entity xnor2 is port
   ( y0       : out std_logic;
     x0, x1 : in    std_logic );
end entity xnor2;

architecture xnor2 of xnor2 is
begin
   y0 <= not (x0 xor x1);
end architecture xnor2;
-------------------------------------
```

Código VHDL 1.9: Puerta XNOR lógica.

```
-------------------------------------
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 1.10: Puerta OR de 2 entradas.

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

entity or3 is
   port ( y0 : out std_logic;
        x0, x1, x2 : in std_logic );
end entity or3;

architecture or3 of or3 is
begin
   y0 <= x0 or x1 or x2;
end architecture or3;
-------------------------------------
```

Código VHDL 1.11: Puerta OR de 3 entradas.

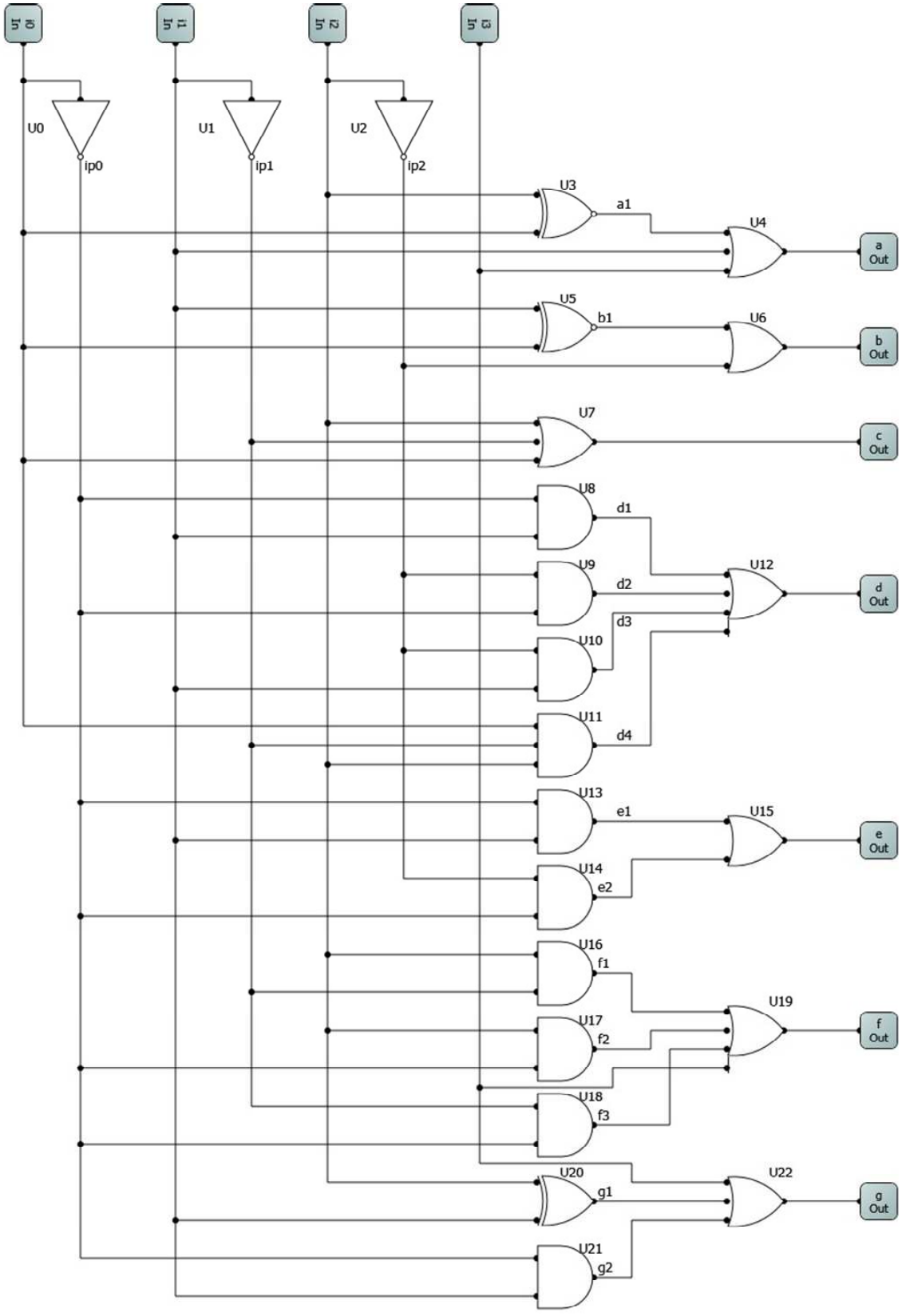


Figura 1.4: Solución al Apartado 2.a: diagrama al nivel de puertas lógicas.

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

entity or4 is
   port ( y0 : out std_logic;
        x0, x1, x2, x3 : in std_logic );
end entity or4;

architecture or4 of or4 is
begin
   y0 <= x0 or x1 or x2 or x3;
end architecture or4;
-------------------------------------
```

Código VHDL 1.12: Puerta OR de 4 entradas.

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

architecture Structural of BCDto7seg is
-- Declaración de las clases de los componentes
   component and2 is
      port (   y0 : out std_logic ;
      x0, x1 : in std_logic);
   end component and2;
   component and3 is
      port (   y0 : out std_logic ;
      x0, x1, x2 : in std_logic);
   end component and3;
   component not1 is
      port ( y0 : out std_logic;
               x0 : in std_logic );
   end component not1;
   component xor2 is
      port ( y0 : out std_logic;
           x0, x1 : in std_logic );
   end component xor2;
   component xnor2 is
      port ( y0 : out std_logic;
           x0, x1 : in std_logic );
   end component xnor2;
   component or2 is
      port ( y0 : out std_logic;
           x0, x1 : in std_logic );
   end component or2;
   component or3 is
      port ( y0 : out std_logic;
           x0, x1, x2 : in std_logic );
   end component or3;
    component or4 is
      port ( y0 : out std_logic;
           x0, x1, x2, x3 : in std_logic );
   end component or4;
```

Código VHDL 1.13: Solución al Apartado 2.b: **architecture** del circuito conversor describiendo su estructura.

```
   signal ip0,ip1,ip2,a1,b1,d1,d2,d3,d4,e1,e2,f1,f2,f3,g1,g2: std_logic;
begin
   U0: not1  port map(ip0, i0);
   U1: not1  port map(ip1, i1);
   U2: not1  port map(ip2, i2);
   U3: xnor2  port map(a1, i2, i0);
   U4: or3  port map(a, i3, i1, a1);
   U5: xnor2  port map(b1, i1, i0);
   U6: or2  port map(b, ip2, b1);
   U7: or3  port map(c, i2, ip1, i0);
   U8: and2  port map(d1, i1, ip0);
   U9: and2  port map(d2, ip2, ip0);
   U10: and2  port map(d3, ip2, i1);
   U11: and3  port map(d4, i2, ip1, i0);
   U12: or4  port map(d, d1, d2, d3, d4);
   U13: and2  port map(e1, i1, ip0);
   U14: and2  port map(e2, ip2, ip0);
   U15: or2  port map(e, e1, e2);
   U16: and2  port map(f1, i2, ip1);
   U17: and2  port map(f2, i2, ip0);
   U18: and2  port map(f3, ip1, ip0);
   U19: or4  port map(f, i3, f1, f2, f3);
   U20: xor2  port map(g1, i2, i1);
   U21: and2  port map(g2, i1, ip0);
   U22: or3  port map(g, i3, g1, g2);
end Structural;
```

Código VHDL 1.14: Solución al Apartado 2.b: continuación de la **architecture** del circuito conversor describiendo su estructura.

```
--------------------------------------------------
-- Banco de pruebas del decodificador BCD a 7 segmentos
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_bcdTo7Seg is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_bcdTo7Seg;

architecture bp_bcdTo7Seg of bp_bcdTo7Seg is
    signal segs : std_logic_vector(6 downto 0);
    signal bcd  : std_logic_vector(3 downto 0);

    component BCDto7seg is
      port ( a, b, c, d, e, f, g : out std_logic;
                i3, i2, i1, i0 : in std_logic );
      end component BCDto7seg;
   -- Procedure que calcula la salida (expected_s) y lo compara con el
   -- valor de salida que se pasa como argumento (actual_s)
   -- Si ambos valores no coinciden, se muestra un mensaje y se
   -- incrementa el contador de errores (error_count)
   procedure check_salida
       ( bcd      : in std_logic_vector(3 downto 0);
         segs         : in std_logic_vector(6 downto 0);
         error_count: inout integer ) is
     variable segs_esp : std_logic_vector(6 downto 0);
   begin
       if (bcd = "0000") then segs_esp := "1111110";
         elsif (bcd = "0001") then   segs_esp := "0110000";
         elsif (bcd = "0010") then segs_esp := "1101101";
         elsif (bcd = "0011") then segs_esp := "1111001";
         elsif (bcd = "0100") then segs_esp := "0110011";
         elsif (bcd = "0101") then segs_esp := "1011011";
         elsif (bcd = "0110") then segs_esp := "1011111";
         elsif (bcd = "0111") then segs_esp := "1110000";
         elsif (bcd = "1000") then segs_esp := "1111111";
         elsif (bcd = "1001") then segs_esp := "1110011";
         else segs_esp := "0000000";
       end if;
```

Código VHDL 1.15: Solución al Apartado 2.c: banco de pruebas del circuito conversor.

```
          assert( segs = segs_esp)
         report "ERROR. Entrada: " & std_logic'image(bcd(3))
                  & std_logic'image(bcd(2)) & std_logic'image(bcd(1)) &
                  std_logic'image(bcd(0)) &
                     ", resultado esperado: " & std_logic'image(segs_esp(6))
              & std_logic'image(segs_esp(5)) & std_logic'image(segs_esp(4)) &
                  std_logic'image(segs_esp(3)) & std_logic'image(segs_esp(2)) &
                  std_logic'image(segs_esp(1)) & std_logic'image(segs_esp(0)) &
                  ", resultado actual: " & std_logic'image(segs(6))
                  & std_logic'image(segs(5)) & std_logic'image(segs(4)) &
                    std_logic'image(segs(3)) & std_logic'image(segs(2)) &
                    std_logic'image(segs(1)) & std_logic'image(segs(0)) &
                  " en el instante  "&
                  time'image(now);

     if (segs /= segs_esp) then
        error_count := error_count + 1;
     end if;

  end procedure check_salida;
  -- Fin de la definición del procedure


begin
    UUT : component BCDto7seg port map
          (segs(6), segs(5), segs(4), segs(3), segs(2),
           segs(1), segs(0), bcd(3), bcd(2), bcd(1), bcd(0));

vec_test : process is
      variable valor  : unsigned (3 downto 0);
      variable error_count : integer := 0;
    begin
      report "Comienza la simulación";
      -- Generar todos los posibles valores de entrada
      for i in 0 to 9 loop
        valor := to_unsigned(i,4);
        bcd <= std_logic(valor(3))&std_logic(valor(2))
        &std_logic(valor(1))&std_logic(valor(0));
        wait for DELAY;
        check_salida(bcd, segs, error_count);
      end loop;

     report "Simulación finalizada con  "& integer'image(error_count) &
" errores";
      wait;     -- Final de la simulación
  end process vec_test;
end architecture bp_bcdTo7Seg;
-------------------------------------------------
```

Código VHDL 1.16: Solución al Apartado 2.c: continuación del banco de pruebas del circuito conversor.