INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2012

EJERCICIO 1

A continuación se muestran dos funciones lógicas `F` y `G`, que dependen de las cuatro variables `A`, `B`, `C` y `D` de la forma mostrada a continuación:

```
F = A B C + C D
G = A B C'
```

1.a) (0.5 puntos) Escriba en VHDL la **entity** de un circuito que implemente las dos funciones lógicas. Es decir, que tenga cuatro entradas `A`, `B`, `C` y `D`, y dos salidas `F` y `G`.

1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* de un circuito que implemente las dos funciones lógicas.

1.c) (0.5 puntos) Dibuje el diagrama al nivel de puertas lógicas de un circuito que implemente estas dos funciones. Emplee para ello puertas lógicas AND y OR de dos entradas, y puerta NOT. 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, las salidas 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

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. Obsérvese que en la figura se han señalado las señales `ab`, `abc`, `cd` y `not_c`, que se usarán en la descripción estructural.

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

entity funcLog_F_G is
   port ( F,G       : out  std_logic;
          a,b,c,d   : in   std_logic );
end entity funcLog_F_G;
-------------------------------------
```

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

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

architecture funcLog_F_G_Comp of funcLog_F_G is
begin
   F <= (a and b and c) or (c and d);
   G <= a and b and not c;
end architecture funcLog_F_G_Comp;
-------------------------------------
```

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

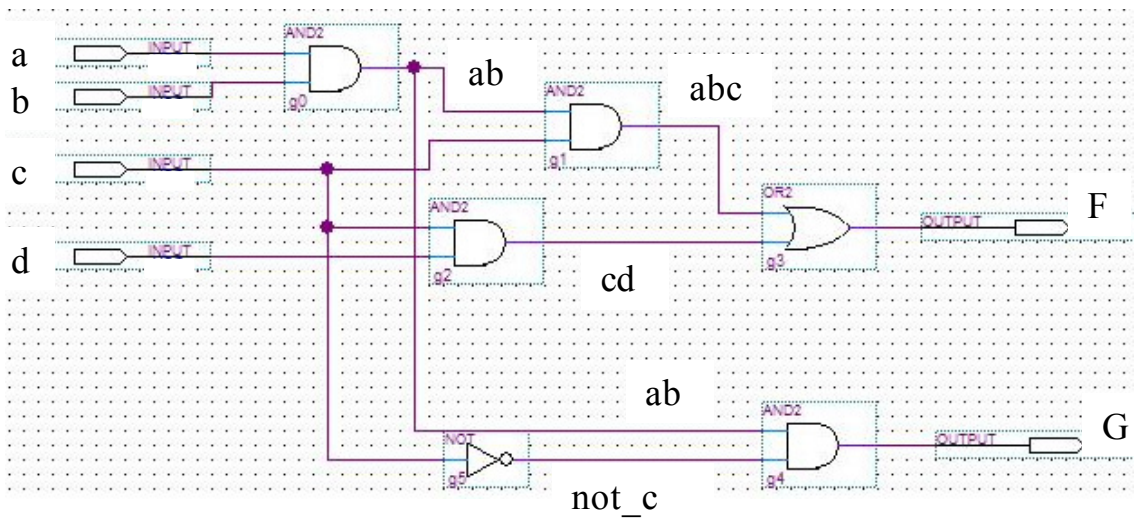

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

El Código VHDL 1.3 – 1.5 es una posible descripción de las puertas lógicas AND y OR de dos entradas, y la puerta NOT.

```
-------------------------------------
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: Solución al Apartado 1.c: puerta AND de dos entradas.

```
-------------------------------------
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: Solución al Apartado 1.c: puerta OR de dos entradas.

```
-------------------------------------
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: Solución al Apartado 1.c: puerta NOT con una entrada.

El Código VHDL 1.6 es una posible descripción estructural del circuito, que corresponde con el diagrama mostrado en la Figura 1.1.

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

architecture funcLog_F_G_Estruc of funcLog_F_G is
   signal not_c, ab, abc, cd: 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 or2 is
      port ( y0          : out std_logic;
             x0, x1      : in  std_logic );
   end component or2;
begin
   -- Instanciación y conexión de los componentes
   g0 : component and2 port map (ab, a, b);
   g1 : component and2 port map (abc, ab, c);
   g2 : component and2 port map (cd, c, d);
   g3 : component or2 port map (F, abc, cd);
   g4 : component and2 port map (G, ab, not_c);
   g5 : component not1 port map (not_c, c);
end architecture funcLog_F_G_Estruc;
-----------------------------------
```

Código VHDL 1.6: Solución al Apartado 1.d: descripción estructural al nivel de puertas lógicas.

Finalmente, el Código VHDL 1.7 es un banco de pruebas para los diseños del circuito que implementa las dos funciones lógicas. En la Figura 1.2 se muestran las formas de onda obtenidas al simular el banco de pruebas con cualquiera de los dos diseños anteriormente realizados. La comprobación de que el diseño funciona correctamente debe hacerse en este caso mediante inspección visual.

```
--------------------------------------
-- 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 x0,x1,x2,x3 : std_logic;  -- Conectar entradas UUT

   component funcLog_F_G is port
      ( F,G         : out std_logic;
        a,b,c,d  : in   std_logic);
   end component funcLog_F_G;

begin
   -- Instanciar y conectar UUT
   uut : component funcLog_F_G port map
         ( F => y0,G => y1,
           a => x0,b => x1,c => x2,d => x3 );
   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
         x3    <=test_in(3);
         x2    <=test_in(2);
         x1    <=test_in(1);
         x0    <=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 1.7: Solución al Apartado 1.e: banco de pruebas.

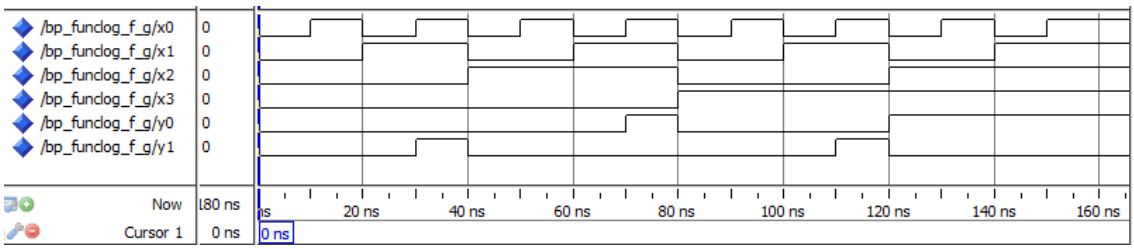

Figura 1.2: Solución al Apartado 1.e: simulación del banco de pruebas.

EJERCICIO 2

2.a) (1.5 puntos) Diseñe en VHDL la **architecture** de un codificador 8 a 3 con prioridad, describiendo el comportamiento del circuito. La **entity** del circuito y la tabla de la verdad se muestran a continuación.

```
entity codif_8a3 is
   port ( codigo  : out std_logic_vector(2 downto 0);
          activo  : out std_logic;
          x       : in  std_logic_vector(7 downto 0) );
end entity codif_8a3;
```

x								codigo			activo
1	–	–	–	–	–	–	–	1	1	1	1
0	1	–	–	–	–	–	–	1	1	0	1
0	0	1	–	–	–	–	–	1	0	1	1
0	0	0	1	–	–	–	–	1	0	0	1
0	0	0	0	1	–	–	–	0	1	1	1
0	0	0	0	0	1	–	–	0	1	0	1
0	0	0	0	0	0	1	–	0	0	1	1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0

2.b) (1.5 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito que ha diseñado en el Apartado 2.a. 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.

2.c) (1.5 puntos) Existen algunas aplicaciones en las cuales se precisa poder modificar la prioridad de las señales de entrada `x(7), ..., x(0)`. Los codificadores que soportan esta funcionalidad se denominan *codificadores con prioridad programable*. Diseñe en VHDL la **architecture** de un codificador 8 a 3 con prioridad programable. La **entity** del circuito es:

```
entity codif_8a3_prog is
   port ( codigo  : out std_logic_vector(2 downto 0);
          activo  : out std_logic;
          x       : in  std_logic_vector(7 downto 0);
          c       : in  std_logic_vector(2 downto 0) );
end entity codif_8a3_prog;
```

El valor de la palabra binaria de tres bits `c(2),c(1),c(0)` especifica cuál de las señales de entrada `x(7), ..., x(0)` tiene mayor prioridad. Por ejemplo, si `c` vale `"011"`, entonces la prioridad es: `x(3),x(2),x(1),x(0), x(7),...,x(4)`.

En la siguiente tabla se indica la prioridad de las señales `x(7),...,x(0)` en función del valor de las señales de control `c(2),c(1),c(0)`.

c	Prioridad							
	<- Mayor							Menor ->
1 1 1	x(7)	x(6)	x(5)	x(4)	x(3)	x(2)	x(1)	x(0)
1 1 0	x(6)	x(5)	x(4)	x(3)	x(2)	x(1)	x(0)	x(7)
1 0 1	x(5)	x(4)	x(3)	x(2)	x(1)	x(0)	x(7)	x(6)
1 0 0	x(4)	x(3)	x(2)	x(1)	x(0)	x(7)	x(6)	x(5)
0 1 1	x(3)	x(2)	x(1)	x(0)	x(7)	x(6)	x(5)	x(4)
0 1 0	x(2)	x(1)	x(0)	x(7)	x(6)	x(5)	x(4)	x(3)
0 0 1	x(1)	x(0)	x(7)	x(6)	x(5)	x(4)	x(3)	x(2)
0 0 0	x(0)	x(7)	x(6)	x(5)	x(4)	x(3)	x(2)	x(1)

2.d) (1.5 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito que ha diseñado en el Apartado 2.c. 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.

Solución al Ejercicio 2

El Código VHDL 1.8 es un posible diseño del codificador 8 a 3 con prioridad. Obsérvese que en dicho diseño se emplea la función `std_match`, que permite comparar vectores con bits don't care. El Código VHDL 1.9 – 1.11 es un banco de pruebas para el codificador. Para testear el circuito debe simularse el banco de pruebas durante 2560 ns.

```
-------------------------------------
-- Codificador 8 a 3 con prioridad
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity codif_8a3 is
   port (  codigo  : out std_logic_vector(2 downto 0);
           activo  : out std_logic;
           x       : in  std_logic_vector(7 downto 0) );
end entity codif_8a3;

architecture codif_8a3 of codif_8a3 is
begin
   codigo <= "111" when std_match(x,"1-------") else
             "110" when std_match(x,"01------") else
             "101" when std_match(x,"001-----") else
             "100" when std_match(x,"0001----") else
             "011" when std_match(x,"00001---") else
             "010" when std_match(x,"000001--") else
             "001" when std_match(x,"0000001-") else
             "000";
   activo <= x(7) or x(6) or x(5) or x(4) or
             x(3) or x(2) or x(1) or x(0);
end architecture codif_8a3;
-------------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.a: codificador 8 a 3 con prioridad.

```
-----------------------------------------------------
-- Banco de pruebas del codificador 8:3 con prioridad
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_codif_8a3_prio is
   constant MAX_COMB : integer := 256;  -- Num. combinac. entrada (2**8)
   constant DELAY    : time    := 10 ns;  -- Retardo usado en el test
end entity bp_codif_8a3_prio;
```

Código VHDL 1.9: Solución al Apartado 2.b: banco de pruebas del codificador 8 a 3 con prioridad (parte inicial).

```
architecture bp_codif_8a3_prio of bp_codif_8a3_prio is
    -- Salidas UUT
    signal activo    : std_logic;
    signal codigo    : std_logic_vector(2 downto 0);
    -- Entradas UUT
    signal x         : std_logic_vector(7 downto 0);

    component codif_8a3 is
        port ( codigo  : out std_logic_vector(2 downto 0);
               activo  : out std_logic;
               x       : in  std_logic_vector(7 downto 0) );
     end component codif_8a3;

begin  -- Cuerpo de la arquitectura
  UUT : component codif_8a3 port map (codigo,activo,x);

  main : process is
    variable esperado_activo   : std_logic;
    variable esperado_codigo   : std_logic_vector(2 downto 0);
    variable error_count       : integer    := 0;
  begin
    report "Comienza la simulación";
    -- Generar todos los posibles valores de entrada
    for i in 0 to (MAX_COMB-1) loop
      x <= std_logic_vector(TO_UNSIGNED(i,8));
      -- Calcular el valor esperado
      if (i=0) then
          esperado_activo := '0';
          esperado_codigo := "000";
      else
          esperado_activo     := '1';
          if     (i=1)    then  esperado_codigo := "000";
          elsif (i<=3)    then  esperado_codigo := "001";
          elsif (i<=7)    then  esperado_codigo := "010";
          elsif (i<=15)   then  esperado_codigo := "011";
          elsif (i<=31)   then  esperado_codigo := "100";
          elsif (i<=63)   then  esperado_codigo := "101";
          elsif (i<=127)  then  esperado_codigo := "110";
          else                   esperado_codigo := "111";
          end if;
      end if;

      wait for DELAY; -- Espera y compara con las salidas de UUT

      if ( esperado_activo /= activo ) then
          report "ERROR en la salida valida. Valor esperado: "  &
                 std_logic'image(esperado_activo)               &
                 ", valor actual: "                             &
                 std_logic'image(activo)                        &
                 " en el instante: "                              &
                 time'image(now);
          error_count := error_count + 1;
      end if;
```

Código VHDL 1.10: Solución al Apartado 2.b: banco de pruebas del codificador 8 a 3 con prioridad (parte intermedia).

```
      if ( esperado_codigo /= codigo ) then
        report "ERROR en la salida codificada. Valor esperado:" &
                  std_logic'image(esperado_codigo(2))           &
                  std_logic'image(esperado_codigo(1))           &
                  std_logic'image(esperado_codigo(0))           &
                  ",valor actual:"                              &
                  std_logic'image(codigo(2))                    &
                  std_logic'image(codigo(1))                    &
                  std_logic'image(codigo(0))                    &
                  " en el instante:"                          &
                  time'image(now);
        error_count := error_count + 1;
      end if;

    end loop;-- Final del bucle for de posibles valores de entrada
    -- Informe del número total de errores
        report  "Hay "                &
                 integer'image(error_count) &
                 " errores.";
    wait;    -- Final de la simulación
  end process main;
end architecture bp_codif_8a3_prio;
-------------------------------------------------
```

Código VHDL 1.11: Solución al Apartado 2.b: banco de pruebas del codificador 8 a 3 con prioridad (parte final).

En el Apartado 2.c se pide diseñar un codificador 8 a 3 con prioridad programable. Una posible forma de realizar el diseño es utilizar ocho codificadores 8 a 3 con prioridad, como el diseñado en el Apartado 2.a, y un multiplexor 8:1, que pase a la salida el código deseado. Esta forma de realizar el diseño es correcta, aunque no demasiado eficiente.

En la Figura 1.3 se muestra un diseño más eficiente. A partir de la señal `c` se genera una señal de máscara de 8 bits, que es usada para poner a cero la parte superior e inferior de la señal `x`. Por ejemplo, si `c` vale `"011"`, entonces la señal `mascara` es `"00001111"`, de modo que:

`arriba_x` vale `"11110000" and x`

y

`abajo_x` vale `"00001111" and x`.

Obsérvese que los bits en `abajo_x` tienen mayor prioridad que los de `arriba_x`, por ello si `abajo_activo` vale `'1'`, se enruta a la salida el código correspondiente a `abajo_x`, es decir, `abajo_codigo`. Si por el contrario la señal `abajo_activo` vale `'0'`, entonces se enruta a la salida el código correspondiente a `arriba_x`, es decir, `arriba_codigo`.

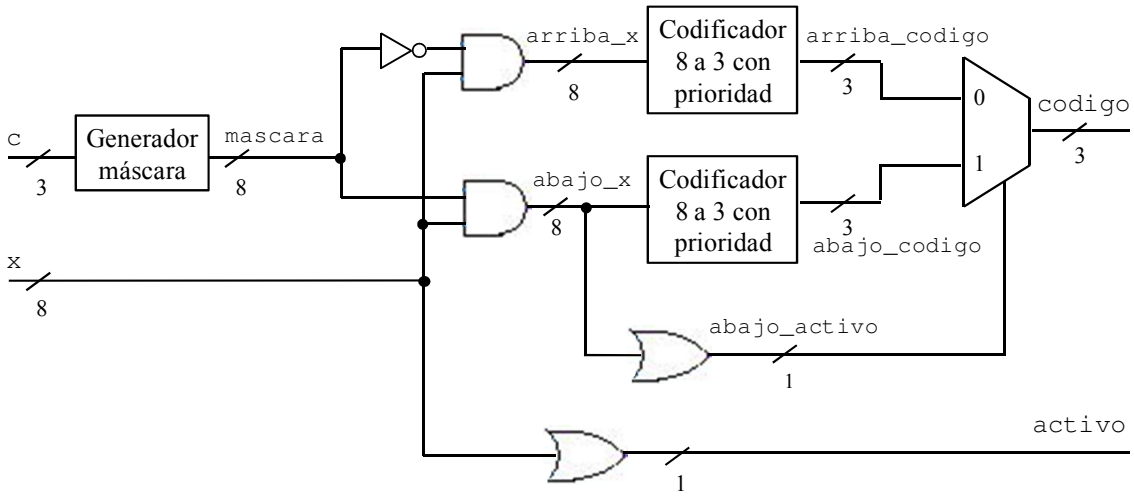

Figura 1.3: Solución al Apartado 2.c: diagrama de bloques de un codificador con prioridad programable.

El Código VHDL 1.12 describe el comportamiento del circuito mostrado en la Figura 1.3. Los dos codificadores 8 a 3 con prioridad usados en el circuito son codificadores como los diseñados en el Apartado 2.a. Obsérvese que en este caso se ha empleado una asignación concurrente condicional para definir el comportamiento de cada uno de ellos.

Finalmente, en el Código VHDL 1.13 – 1.16 se muestra un posible banco de pruebas para el circuito.

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

entity codif_8a3_prog is
   port ( codigo  : out std_logic_vector(2 downto 0);
          activo  : out std_logic;
          x       : in  std_logic_vector(7 downto 0);
          c       : in  std_logic_vector(2 downto 0) );
end entity codif_8a3_prog;

architecture codif_8a3_prog of codif_8a3_prog is
   signal mascara, abajo_x, arriba_x: std_logic_vector(7 downto 0);
   signal abajo_codigo, arriba_codigo: std_logic_vector(2 downto 0);
   signal abajo_activo: std_logic;
begin
   with c select
     mascara <= "00000001" when "000",
                "00000011" when "001",
                "00000111" when "010",
                "00001111" when "011",
                "00011111" when "100",
                "00111111" when "101",
                "01111111" when "110",
                "11111111" when others;
   abajo_x <= x and mascara;
   arriba_x <= x and (not mascara);
   abajo_codigo <= "111" when abajo_x(7) = '1' else
                   "110" when abajo_x(6) = '1' else
                   "101" when abajo_x(5) = '1' else
                   "100" when abajo_x(4) = '1' else
                   "011" when abajo_x(3) = '1' else
                   "010" when abajo_x(2) = '1' else
                   "001" when abajo_x(1) = '1' else
                   "000";
   arriba_codigo <= "111" when arriba_x(7) = '1' else
                    "110" when arriba_x(6) = '1' else
                    "101" when arriba_x(5) = '1' else
                    "100" when arriba_x(4) = '1' else
                    "011" when arriba_x(3) = '1' else
                    "010" when arriba_x(2) = '1' else
                    "001" when arriba_x(1) = '1' else
                    "000";
   abajo_activo <= abajo_x(7) or abajo_x(6) or abajo_x(5) or
                   abajo_x(4) or abajo_x(3) or abajo_x(2) or
                   abajo_x(1) or abajo_x(0);
   codigo <= abajo_codigo when abajo_activo='1'  else
             arriba_codigo;
   activo <= x(7) or x(6) or x(5) or x(4) or
             x(3) or x(2) or x(1) or x(0);
end architecture codif_8a3_prog;
 ---------------------------------------
```

Código VHDL 1.12: Solución al Apartado 2.c: codificador 8 a 3 con prioridad programable.

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

entity bp_codif_8a3_prio_prog is
   constant MAX_COMB : integer := 256;  -- Num. combinac. entrada (2**8)
   constant MAX_PRIO : integer := 8;
   constant DELAY    : time    := 10 ns;  -- Retardo usado en el test
end entity bp_codif_8a3_prio_prog;


architecture bp_codif_8a3_prio_prog of bp_codif_8a3_prio_prog is
    -- Salidas UUT
    signal activo   : std_logic;
    signal codigo   : std_logic_vector(2 downto 0);
    -- Entradas UUT
    signal x        : std_logic_vector(7 downto 0);
    signal c        : std_logic_vector(2 downto 0);

   component codif_8a3_prog is
     port ( codigo  : out std_logic_vector(2 downto 0);
            activo  : out std_logic;
            x       : in  std_logic_vector(7 downto 0);
            c       : in  std_logic_vector(2 downto 0) );
   end component codif_8a3_prog;

begin  -- Cuerpo de la architecture

   UUT : component codif_8a3_prog port map (codigo,activo,x,c);

   main : process is
     variable esperado_activo   : std_logic;
     variable esperado_codigo   : std_logic_vector(2 downto 0);
     variable error_count       : integer    := 0;
     variable x_var: std_logic_vector(7 downto 0);
   begin
     report "Comienza la simulación";
     -- Generar todos los posibles valores de entrada
     for i_c in  (MAX_PRIO-1) downto 0 loop
        c <= std_logic_vector(TO_UNSIGNED(i_c,3));
       for i in 0 to (MAX_COMB-1) loop
         x_var := std_logic_vector(TO_UNSIGNED(i,8));
         x <= std_logic_vector(TO_UNSIGNED(i,8));
```

Código VHDL 1.13: Solución al Apartado 2.d: banco de pruebas del codificador 8 a 3 con prioridad programable (parte inicial).

```
-- Calcular el valor esperado
if (i=0) then
 esperado_activo := '0';
 esperado_codigo := "000";
else
 esperado_activo := '1';
 if (i_c = 7) then
    if  std_match(x_var,"00000001")  then  esperado_codigo := "000";
    elsif std_match(x_var,"0000001-")  then  esperado_codigo := "001";
    elsif std_match(x_var,"000001--") then  esperado_codigo := "010";
    elsif std_match(x_var,"00001---") then  esperado_codigo := "011";
    elsif std_match(x_var,"0001----") then  esperado_codigo := "100";
    elsif std_match(x_var,"001-----") then  esperado_codigo := "101";
    elsif std_match(x_var,"01------") then  esperado_codigo := "110";
    elsif std_match(x_var,"1-------") then  esperado_codigo := "111";
    end if;
 elsif (i_c = 6) then
    if std_match(x_var,"-1------") then esperado_codigo := "110";
    elsif std_match(x_var,"-01-----") then esperado_codigo := "101";
    elsif std_match(x_var,"-001----") then esperado_codigo := "100";
    elsif std_match(x_var,"-0001---") then esperado_codigo := "011";
    elsif std_match(x_var,"-00001--") then esperado_codigo := "010";
    elsif std_match(x_var,"-000001-") then esperado_codigo := "001";
    elsif std_match(x_var,"-0000001") then esperado_codigo := "000";
    elsif std_match(x_var,"10000000") then esperado_codigo := "111";
    else                  esperado_codigo := "000";
    end if;
 elsif (i_c = 5) then
    if  std_match(x_var,"--1-----") then esperado_codigo := "101";
    elsif std_match(x_var,"--01----") then esperado_codigo := "100";
    elsif std_match(x_var,"--001---") then esperado_codigo := "011";
    elsif std_match(x_var,"--0001--") then esperado_codigo := "010";
    elsif std_match(x_var,"--00001-") then esperado_codigo := "001";
    elsif std_match(x_var,"--000001") then esperado_codigo := "000";
    elsif std_match(x_var,"1-000000")  then  esperado_codigo := "111";
    elsif std_match(x_var,"01000000") then  esperado_codigo := "110";
    else                  esperado_codigo := "000";
    end if;
 elsif (i_c = 4) then
    if  std_match(x_var,"---1----") then esperado_codigo := "100";
    elsif std_match(x_var,"---01---") then esperado_codigo := "011";
    elsif std_match(x_var,"---001--") then esperado_codigo := "010";
    elsif std_match(x_var,"---0001-") then esperado_codigo := "001";
    elsif std_match(x_var,"---00001") then esperado_codigo := "000";
    elsif std_match(x_var,"1--00000") then esperado_codigo := "111";
    elsif std_match(x_var,"01-00000")  then  esperado_codigo := "110";
    elsif std_match(x_var,"00100000") then  esperado_codigo := "101";
    else                  esperado_codigo := "000";
    end if;
```

Código VHDL 1.14: Solución al Apartado 2.d: banco de pruebas del codificador 8 a 3 con prioridad programable (continuación).

```
  elsif (i_c = 3) then
    if  std_match(x_var,"----1---") then esperado_codigo := "011";
    elsif std_match(x_var,"----01--") then esperado_codigo := "010";
    elsif std_match(x_var,"----001-") then esperado_codigo := "001";
    elsif std_match(x_var,"----0001") then esperado_codigo := "000";
    elsif std_match(x_var,"1---0000") then esperado_codigo := "111";
    elsif std_match(x_var,"01--0000") then esperado_codigo := "110";
    elsif std_match(x_var,"001-0000")  then  esperado_codigo := "101";
    elsif std_match(x_var,"00010000") then  esperado_codigo := "100";
    else                  esperado_codigo := "000";
    end if;
  elsif (i_c = 2) then
    if  std_match(x_var,"-----1--") then esperado_codigo := "010";
    elsif std_match(x_var,"-----01-") then esperado_codigo := "001";
    elsif std_match(x_var,"-----001") then esperado_codigo := "000";
    elsif std_match(x_var,"1----000") then esperado_codigo := "111";
    elsif std_match(x_var,"01---000") then esperado_codigo := "110";
    elsif std_match(x_var,"001--000") then esperado_codigo := "101";
    elsif std_match(x_var,"0001-000")  then  esperado_codigo := "100";
    elsif std_match(x_var,"00001000") then  esperado_codigo := "011";
    else                  esperado_codigo := "000";
    end if;
  elsif (i_c = 1) then
    if  std_match(x_var,"------1-") then esperado_codigo := "001";
    elsif std_match(x_var,"------01") then esperado_codigo := "000";
    elsif std_match(x_var,"1-----00") then esperado_codigo := "111";
    elsif std_match(x_var,"01----00") then esperado_codigo := "110";
    elsif std_match(x_var,"001---00") then esperado_codigo := "101";
    elsif std_match(x_var,"0001--00") then esperado_codigo := "100";
    elsif std_match(x_var,"00001-00")  then  esperado_codigo := "011";
    elsif std_match(x_var,"00000100") then  esperado_codigo := "010";
    else                  esperado_codigo := "000";
    end if;
  elsif (i_c = 0) then
    if std_match(x_var,"-------1") then esperado_codigo := "000";
    elsif std_match(x_var,"1------0") then esperado_codigo := "111";
    elsif std_match(x_var,"01-----0") then esperado_codigo := "110";
    elsif std_match(x_var,"001----0") then esperado_codigo := "101";
    elsif std_match(x_var,"0001---0") then esperado_codigo := "100";
    elsif std_match(x_var,"00001--0") then esperado_codigo := "011";
    elsif std_match(x_var,"000001-0")  then  esperado_codigo := "010";
    elsif std_match(x_var,"00000010") then  esperado_codigo := "001";
    else                  esperado_codigo := "000";
    end if;
  end if;
end if;
```

Código VHDL 1.15: Solución al Apartado 2.d: banco de pruebas del codificador 8 a 3 con prioridad programable (continuación).

```
      wait for DELAY; -- Espera y compara con las salidas de UUT
      if ( esperado_activo /= activo )  then
        report "ERROR en la salida valida. Valor esperado:"  &
                    std_logic'image(esperado_activo)                &
                    ",valor actual:"                                &
                    std_logic'image(activo)                         &
                    " en el instante:"                               &
                    time'image(now);
        error_count := error_count + 1;
      end if;

      if ( esperado_codigo /= codigo )  then
        report "ERROR en la salida codificada. Valor esperado:" &
                    std_logic'image(esperado_codigo(2))          &
                    std_logic'image(esperado_codigo(1))          &
                    std_logic'image(esperado_codigo(0))          &
                    ",valor actual:"                             &
                    std_logic'image(codigo(2))                   &
                    std_logic'image(codigo(1))                   &
                    std_logic'image(codigo(0))                  &
                    " en el instante:"                           &
                    time'image(now);
        error_count := error_count + 1;
      end if;

   end loop; -- Final del bucle for de posibles valores de entrada
  end loop;

  -- Informe del número total de errores
  report   "Hay "              &
           integer'image(error_count) &
           " errores.";

  wait;    -- Final de la simulación

end process main;

end architecture bp_codif_8a3_prio_prog;
-------------------------------------------------
```

Código VHDL 1.16: Solución al Apartado 2.d: banco de pruebas del codificador 8 a 3 con prioridad programable (parte final).