INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2016

EJERCICIO 1

Se desea diseñar un circuito digital que implemente las funciones `F` y `G` cuya tabla de verdad se muestra a continuación, que dependen de las tres variables `x`, `y` y `z`:

x	y	z	F	G
'0'	'0'	'0'	'0'	'0'
'0'	'0'	'1'	'0'	'1'
'0'	'1'	'0'	'0'	'1'
'0'	'1'	'1'	'1'	'1'
'1'	'0'	'0'	'0'	'0'
'1'	'0'	'1'	'0'	'0'
'1'	'1'	'0'	'0'	'1'
'1'	'1'	'1'	'1'	'1'

1.a) (0.5 puntos) Obtenga las funciones lógicas `F` y `G` a partir de la tabla de verdad. Escriba en VHDL la **entity** del circuito que implemente las dos funciones lógicas. Es decir, que tenga tres entradas `x`, `y` y `z`, y dos salidas `F` y `G`.

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

1.c) (1 punto) Dibuje el diagrama de un circuito que implemente estas dos funciones lógicas al nivel de puertas lógicas. No es necesario que el circuito esté simplificado. 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) (0.5 puntos) 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 el cronograma obtenido al realizar la simulación del banco de pruebas.

Solución al Ejercicio 1

La **entity** del circuito que implementa las dos funciones lógicas se muestra en Código VHDL 1.1. El Código VHDL 1.2 muestra la **architecture** del circuito describiendo su comportamiento.

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

entity funcFG is
   port ( F, G    : out std_logic;
          x, y, z  : in std_logic );
end entity funcFG;
-------------------------------------
```

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

```
-------------------------------------
architecture Comp of funcFG is
begin
 F <= y and z;
 G <= y or (not x and z);
end architecture Comp;
-------------------------------------
```

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

La Figura 1.1 muestra el diagrama del circuito implementado empleando dos puertas AND de dos entradas, una puerta OR de dos entradas y un inversor.

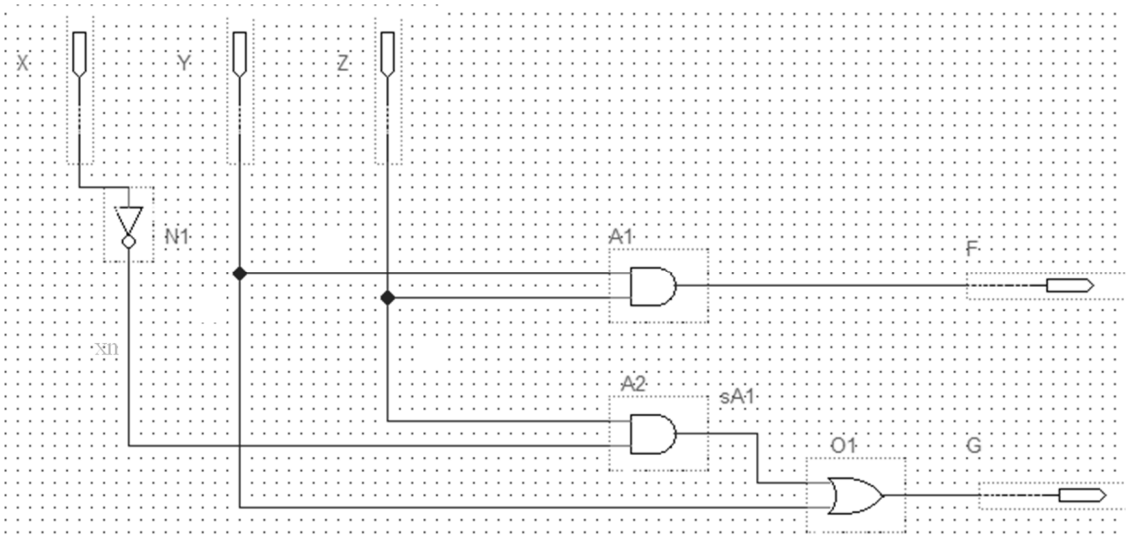

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

El código VHDL de la **entity** y la **architecture** de estas tres puertas lógicas se muestra en Código VHDL 1.3, 1.4 y 1.5, respectivamente. El Código VHDL 1.6 muestra la **architecture** del circuito describiendo su estructura.

```
----------------------------------
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.

```
-------------------------------------
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: Puerta OR 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 circuito_Estruc of funcFG is
   signal xn: std_logic;
   signal sA1: 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
   N1 : component not1 port map (xn, x);
   A1 : component and2 port map (F, y, z);
   A2 : component and2 port map (sA1, xn, z);
   O1 : component or2 port map (G, y, sA1);
end architecture circuito_Estruc;
-----------------------------------
```

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

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.7. El cronograma obtenido al simular el banco de pruebas es mostrado en la Figura 1.2.

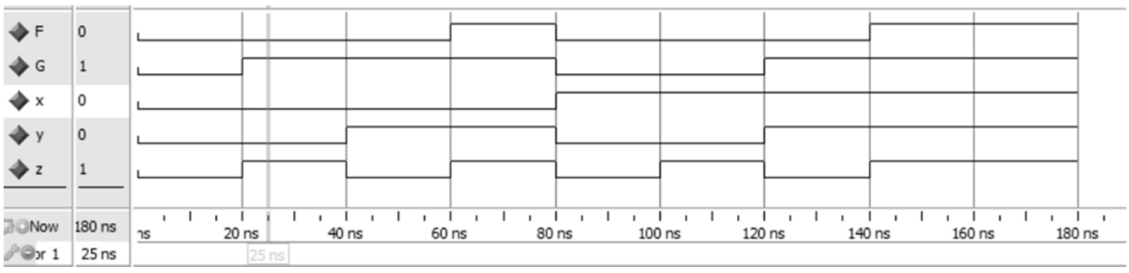

Figura 1.2: Cronograma del Apartado 1.e.

```
--------------------------------------------------
-- Banco de pruebas del circuito Ejercicio 1
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_funcFG is
    constant DELAY      : time      := 20 ns; -- Retardo usado en el test
end entity bp_funcFG;

architecture bp_funcFG of bp_funcFG is
    signal F, G : std_logic;
    signal x, y, z   : std_logic;

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

begin
    UUT : component funcFG port map
          (F, G, x, y, z);

    vec_test : process is
      variable valor  : unsigned (2 downto 0);
    begin
      -- Generar todos los posibles valores de entrada
      for i in 0 to 7 loop
        valor := to_unsigned(i,3);
        x <= std_logic(valor(2));
        y <= std_logic(valor(1));
        z <= std_logic(valor(0));
        wait for DELAY;
      end loop;
      wait;     -- Final de la simulación
    end process vec_test;
end architecture bp_funcFG;
--------------------------------------------------
```

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

EJERCICIO 2

2.a) (1 punto) Se pretende diseñar un codificador de prioridad 4 a 2 empleando únicamente puertas AND de dos entradas, puertas OR de 2 entradas e inversores.

El circuito codificador ha de tener una señal de entrada `x` de 4 bits y dos señales de salida, `codigo` de 2 bits y `activo` de 1 bit. El comportamiento del circuito viene descrito por la siguiente tabla de verdad:

x(3:0)	codigo(1:0)	activo
0000	00	'0'
0001	00	'1'
001 -	01	'1'
01 - -	10	'1'
1 - - -	11	'1'

Dibuje el diagrama del circuito codificador 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.

Finalmente, escriba en VHDL una **entity** y una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

2.b) (1 punto) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito *codificador* diseñado en el apartado anterior. 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.a. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas.

2.c) (3 puntos) Escriba en VHDL la **architecture** que describe la estructura de un circuito codificador de prioridad de 8 a 3, empleando para ello únicamente codificadores de prioridad 4 a 2 iguales a los diseñados en el Apartado 2.a y puertas lógicas (AND de dos entradas, OR de dos entradas e inversores). Dibuje el diagrama circuital correspondiente a dicho diseño. La tabla de verdad y la **entity** del circuito codificador de prioridad de 8 a 3 se muestran a continuación.

x(7:0)	codigo(2:0)	activo
00000000	000	'0'
00000001	000	'1'
0000001 -	001	'1'
000001 - -	010	'1'
00001 - - -	011	'1'
0001 - - - -	100	'1'
001 - - - - -	101	'1'
01 - - - - - -	110	'1'
1 - - - - - - -	111	'1'

```
entity codificadorP8a3 is
  port ( codigo: out std_logic_vector(2 downto 0);
         activo: out std_logic;
         x : in std_logic_vector(7 downto 0) );
end entity codificadorP8a3;
```

2.d) (1 punto) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito *codificador* 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. Emplee este banco de pruebas para comprobar el diseño realizado al contestar el Apartado 2.c. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas.

Solución al Ejercicio 2

El diagrama del codificador de prioridad 4 a 2 se muestra en la Figura 1.3.

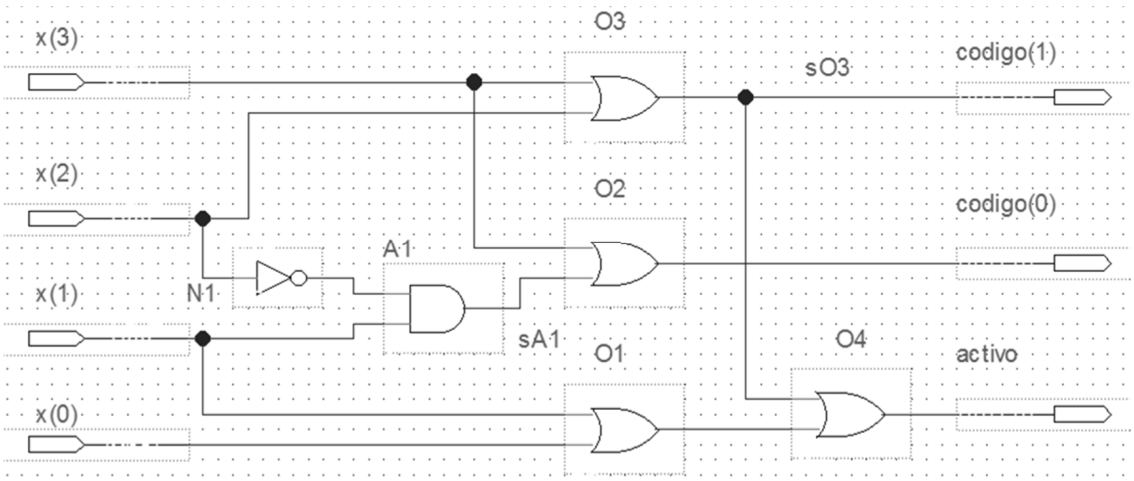

Figura 1.3: Diagrama circuital del codificador de prioridad 4 a 2.

El código VHDL de la **entity** y la **architecture** de las puertas lógicas se muestra en Código VHDL 1.3, 1.4 y 1.5, respectivamente. El Código VHDL 1.8 muestra el código VHDL del circuito describiendo estructura.

El banco de pruebas del codificador de prioridad 4 a 2 se muestra en Código VHDL 1.9–1.10. El cronograma obtenido al simular el banco de pruebas se muestra en la Figura 1.4.

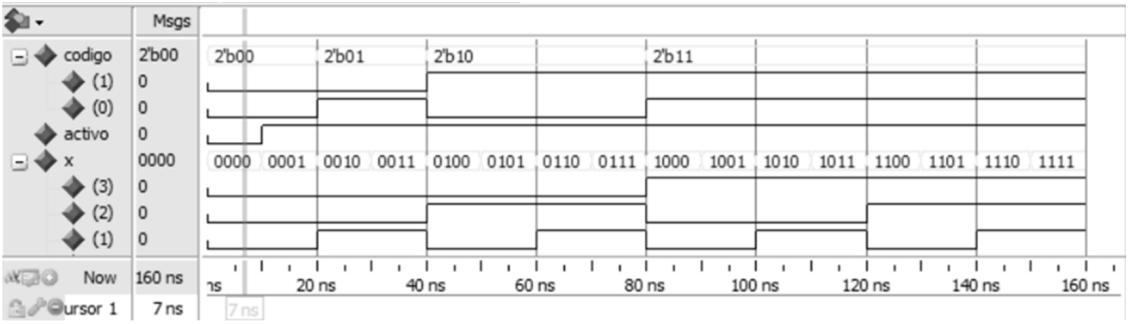

Figura 1.4: Cronograma del Apartado 2.b.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity codificadorP4a2 is
   port ( codigo: out std_logic_vector(1 downto 0);
          activo: out std_logic;
          x : in std_logic_vector(3 downto 0) );
end entity codificadorP4a2;
architecture circuito_Estruc of codificadorP4a2 is
   signal sA1, sO1, sO3, xn2: 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
   N1 : component not1 port map (xn2, x(2));
   A1 : component and2 port map (sA1, xn2, x(1));
   O1 : component or2 port map (sO1, x(0), x(1));
   O2 : component or2 port map (codigo(0), x(3), sA1);
   O3 : component or2 port map (sO3, x(2), x(3));
   O4 : component or2 port map (activo, sO3, sO1);
   codigo(1)<=sO3;
end architecture circuito_Estruc;
---------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.a: codificador de prioridad 4 a 2.

```
  ----------------------------------------------------
-- Banco de pruebas del codificador de prioridad 4 a 2
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_codificadorP4a2 is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_codificadorP4a2;

architecture bp_codificadorP4a2 of bp_codificadorP4a2 is
    signal codigo : std_logic_vector(1 downto 0);
    signal activo  : std_logic;
    signal x: std_logic_vector (3 downto 0);

    component codificadorP4a2 is
    port ( codigo: out std_logic_vector(1 downto 0);
           activo: out std_logic;
           x : in std_logic_vector(3 downto 0) );
    end component codificadorP4a2;
   -- Procedure que calcula la salida  y lo compara con el
   -- valor de salida que se pasa como argumento
   -- Si ambos valores no coinciden, se muestra un mensaje y se
   -- incrementa el contador de errores (error_count)
   procedure check_salida
       ( x      : in std_logic_vector(3 downto 0);
         codigo         : in std_logic_vector(1 downto 0);
         activo        : in std_logic;
         error_count: inout integer ) is
         variable codigo_esp: std_logic_vector(1 downto 0);
         variable activo_esp: std_logic;
   begin
        if std_match  (x,"0000") then codigo_esp := "00";
        elsif std_match(x,"0001") then  codigo_esp := "00";
        elsif std_match(x,"001-")   then codigo_esp := "01";
        elsif std_match(x,"01--") then  codigo_esp := "10";
        elsif std_match(x,"1---") then codigo_esp := "11";
        else codigo_esp := "00";
        end if;
        if (x = "0000") then activo_esp := '0';
        else activo_esp:= '1';
        end if;
```

Código VHDL 1.9: Solución al Apartado 2.b: banco de pruebas del codificador de prioridad 4 a 2.

```
            assert( codigo = codigo_esp)
          report "ERROR. Entrada: " & std_logic'image(x(3))
         & std_logic'image(x(2)) & std_logic'image(x(1)) &
          std_logic'image(x(0)) &", resultado esperado: " &
           std_logic'image(codigo_esp(1)) & std_logic'image(codigo_esp(0)) &
           ", resultado actual: " &
          std_logic'image(codigo(1)) & std_logic'image(codigo(0)) &
                  " en el instante " &
                  time'image(now);
           assert( activo = activo_esp)
          report "ERROR. Entrada: " & std_logic'image(x(3))
                  & std_logic'image(x(2)) & std_logic'image(x(1)) &
                  std_logic'image(x(0)) &
                  ", resultado esperado: " &
                  std_logic'image(activo_esp)  &
                  ", resultado actual: " &
                   std_logic'image(activo) &
                  " en el instante " &
                  time'image(now);

      if (codigo /= codigo_esp)or (activo /= activo_esp) then
        error_count := error_count + 1;
      end if;

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


begin
    UUT : component codificadorP4a2 port map
          (codigo, activo, x);

vec_test : process is
       variable error_count : integer := 0;
     begin
       report "Comienza la simulación";
       -- Generar todos los posibles valores de entrada
       for i in 0 to 15 loop
          x <= std_logic_vector(to_unsigned(i,4));
          wait for DELAY;
          check_salida(x, codigo, activo, 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_codificadorP4a2;
-------------------------------------------------
```

Código VHDL 1.10: Continuación del banco de pruebas del codificador de prioridad 4 a 2.

El diagrama circuital del codificador de prioridad 8 a 3 está en la Figura 1.5. El diseño del codificador de prioridad 8 a 3 siguiendo el diagrama de la anterior figura, se muestra en Código VHDL 1.11. El banco de pruebas de este codificador está en Código VHDL 1.12–1.13. El cronograma obtenido al simular dicho banco de pruebas se puede ver en la Figura 1.6.

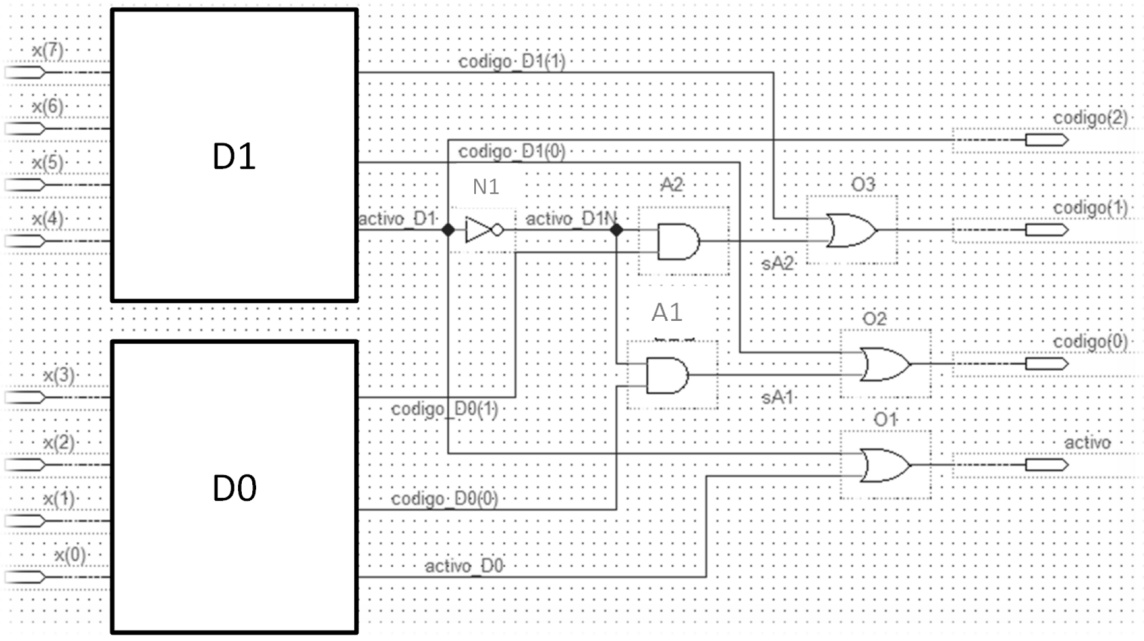

Figura 1.5: Diagrama circuital del codificador de prioridad de 8 a 3.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity codificadorP8a3 is
   port ( codigo: out std_logic_vector(2 downto 0);
            activo: out std_logic;
            x : in std_logic_vector(7 downto 0) );
end entity codificadorP8a3;
architecture circuito_Estruc of codificadorP8a3 is
   signal sA2, sA1, activo_D1N: std_logic;
   signal activo_D1, activo_D0: std_logic;
   signal codigo_D0, codigo_D1: std_logic_vector(1 downto 0);
-- Declaración de las clases de los componentes
     component codificadorP4a2 is
     port ( codigo: out std_logic_vector(1 downto 0);
            activo: out std_logic;
            x : in std_logic_vector(3 downto 0) );
     end component codificadorP4a2;

   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
   D1 : component codificadorP4a2 port map (codigo_D1, activo_D1, x(7
downto 4));
   D0 : component codificadorP4a2 port map (codigo_D0, activo_D0, x(3
downto 0));
   N1 : component not1 port map (activo_D1N, activo_D1);
   A2 : component and2 port map (sA2, activo_D1N, codigo_D0(1));
   A1 : component and2 port map (sA1, activo_D1N, codigo_D0(0));
   O1 : component or2 port map (activo, activo_D1, activo_D0);
   O2 : component or2 port map (codigo(0), sA1, codigo_D1(0));
   O3 : component or2 port map (codigo(1), sA2, codigo_D1(1));
   codigo(2)<=activo_D1;
end architecture circuito_Estruc;
---------------------------------
```

Código VHDL 1.11: Solución al Apartado 2.c: Diseño del codificador de prioridad 8 a 3.

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

entity bp_codificadorP8a3 is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_codificadorP8a3;

architecture bp_codificadorP8a3 of bp_codificadorP8a3 is
    signal codigo : std_logic_vector(2 downto 0);
    signal activo   : std_logic;
    signal x: std_logic_vector (7 downto 0);

    component codificadorP8a3 is
    port ( codigo: out std_logic_vector(2 downto 0);
           activo: out std_logic;
           x : in std_logic_vector(7 downto 0) );
    end component codificadorP8a3;
   -- Procedure que calcula la salida y lo compara con el
   -- valor de salida que se pasa como argumento
   -- Si ambos valores no coinciden, se muestra un mensaje y se
   -- incrementa el contador de errores (error_count)
   procedure check_salida
       ( x      : in std_logic_vector(7 downto 0);
         codigo         : in std_logic_vector(2 downto 0);
         activo       : in std_logic;
         error_count: inout integer ) is
         variable codigo_esp: std_logic_vector(2 downto 0);
         variable activo_esp: std_logic;
   begin
        if std_match(x,"00000000" ) then codigo_esp := "000";
        elsif std_match(x,"00000001" ) then  codigo_esp := "000";
        elsif std_match(x,"0000001-" )  then codigo_esp := "001";
        elsif std_match(x,"000001--" ) then codigo_esp := "010";
        elsif std_match(x,"00001---"  ) then codigo_esp := "011";
        elsif std_match(x,"0001----" ) then codigo_esp := "100";
        elsif std_match(x,"001-----")  then codigo_esp := "101";
        elsif std_match(x,"01------")  then codigo_esp := "110";
        elsif std_match(x,"1-------" )  then codigo_esp := "111";
        else codigo_esp := "00";
        end if;
        if (x = "00000000") then activo_esp := '0';
        else activo_esp:= '1';
        end if;
```

Código VHDL 1.12: Solución al Apartado 2.d: banco de pruebas del codificador de prioridad 8 a 3.

```
            assert( codigo = codigo_esp)
          report "ERROR. Entrada: " & std_logic'image(x(7))
                    & std_logic'image(x(6)) & std_logic'image(x(5)) &
                    std_logic'image(x(4)) & std_logic'image(x(3))
                    & std_logic'image(x(2)) & std_logic'image(x(1)) &
                    std_logic'image(x(0)) &", resultado esperado: " &
                    std_logic'image(codigo_esp(2)) &std_logic'image(codigo_esp(0))
&                        ", resultado actual: "
                    &   std_logic'image(codigo(2)) &
                     std_logic'image(codigo(1))   &   std_logic'image(codigo(0))   &

                    " en el instante "
                    & time'image(now);
           assert( activo = activo_esp)
          report "ERROR. Entrada: " & std_logic'image(x(7))
                    & std_logic'image(x(6)) & std_logic'image(x(5)) &
                    std_logic'image(x(4)) & std_logic'image(x(3))
                    & std_logic'image(x(2)) & std_logic'image(x(1)) &
                    std_logic'image(x(0)) &
                    ", resultado esperado: " &
                    std_logic'image(activo_esp)   &
                    ", resultado actual: " &
                     std_logic'image(activo) &
                    " en el instante " &
                    time'image(now);

      if (codigo /= codigo_esp)or (activo /= activo_esp) then
        error_count := error_count + 1;
      end if;

   end procedure check_salida;
   -- Fin de la definición del procedure
begin
    UUT : component codificadorP8a3 port map
          (codigo, activo, x);

vec_test : process is
       variable error_count : integer := 0;
     begin
       report "Comienza la simulación";
       -- Generar todos los posibles valores de entrada
       for i in 0 to 255 loop
          x <= std_logic_vector(to_unsigned(i,8));
          wait for DELAY;
          check_salida(x, codigo, activo, 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_codificadorP8a3;
-------------------------------------------------
```

Código VHDL 1.13: Solución al Apartado 2.d: continuación del banco de pruebas del codificador de prioridad 8 a 3.

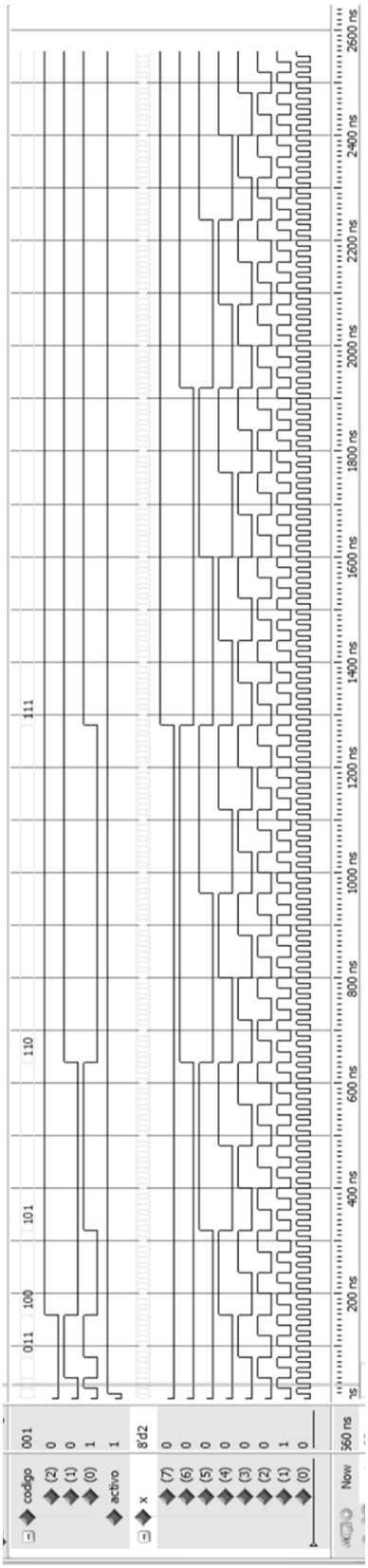

Figura 1.6: Cronograma del Apartado 2.d.