INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2016

EJERCICIO 1

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

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

1.a) (0.25 puntos) Obtenga las función lógica `F` a partir de la tabla de verdad. Escriba en VHDL la **entity** del circuito que implemente la función lógica. Es decir, que tenga tres entradas `x`, `y` y `z`, y una salida `F`.

1.b) (0.75 puntos) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

1.c) (0.25 punto) Dibuje el diagrama de un circuito que implemente esta función lógica 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) (0.75 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.

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 para comprobar los circuitos diseñado en los Apartados 1.b y 1.d.

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

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

```
-------------------------------------
architecture Comp of funcF is
begin
--F = xy +xz
 F <= (x and y) or (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 y una puerta OR de dos entradas.

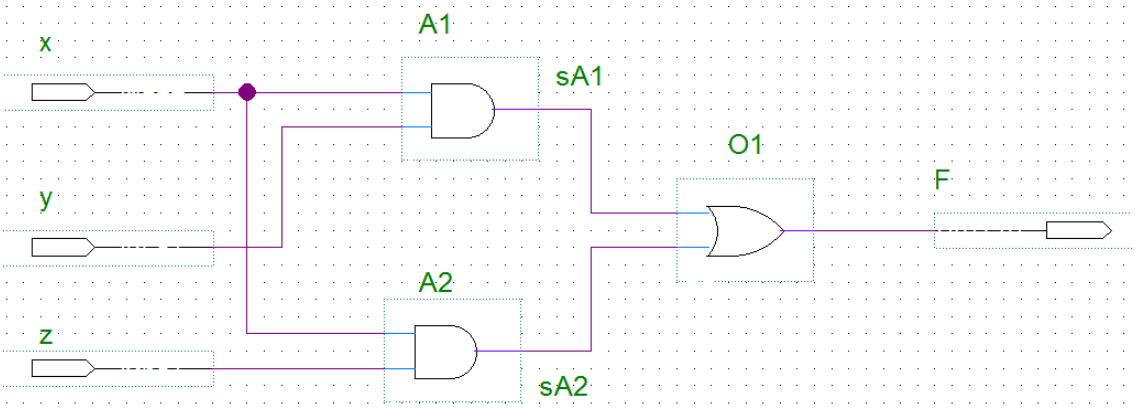

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

El código VHDL de la **entity** y la **architecture** de estas dos puertas lógicas se muestra en Código VHDL 1.3 y 1.4, respectivamente. El Código VHDL 1.5 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;
architecture circuito_Estruc of funcF is
--F = xy +xz
   signal sA1, sA2: 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 or2 is
      port ( y0 : out std_logic;
           x0, x1 : in std_logic );
   end component or2;
begin
-- Instanciación y conexión de los componentes
   A1 : component and2 port map (sA1, x, y);
   A2 : component and2 port map (sA2, x, z);
   O1 : component or2 port map (F, sA1, sA2);
end architecture circuito_Estruc;
-----------------------------------
```

Código VHDL 1.5: 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 1.6. Los dos cronogramas obtenidos al simular el banco de pruebas aplicado al diseño del Apartado 1.b. y 1.d se muestran, respectivamente, en las Figuras 1.2 y 1.3.

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

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

architecture bp_funcF of bp_funcF is
    signal F : std_logic;
    signal x, y, z   : std_logic;
    component funcF is
      port (   F : out std_logic;
               x, y, z : in std_logic );
      end component funcF;
begin
    UUT : component funcF port map
          (F, 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_funcF;
-------------------------------------------------
```

Código VHDL 1.6: Solución al Apartado 1.e: banco de pruebas del circuito.

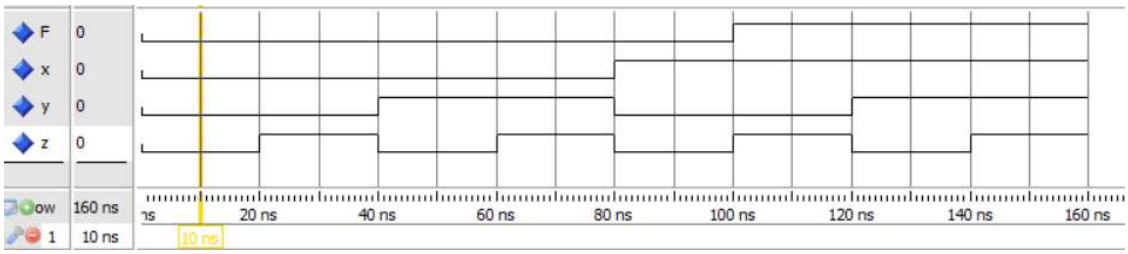

Figura 1.2: Cronograma del banco de pruebas aplicado al diseño del Apartado 1.b.

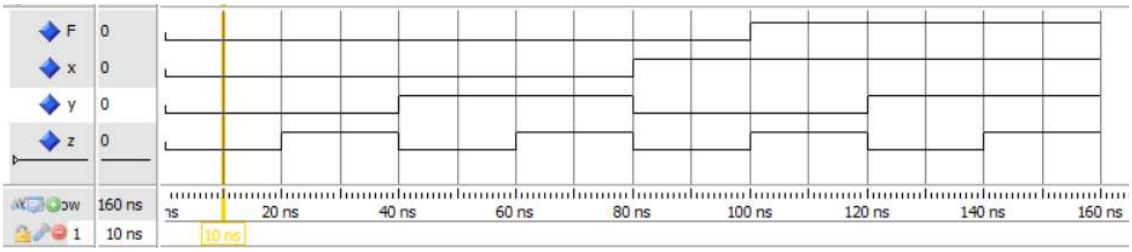

Figura 1.3: Cronograma del banco de pruebas aplicado al diseño del Apartado 1.d.

EJERCICIO 2

2.a) (0.5 punto) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de un codificador de prioridad 4 a 2.

El circuito codificador ha de tener una señal de entrada `r` 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:

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

Realice el diseño del circuito empleando una sentencia **if** para generar la señal `activo` y una sentencia **case** para generar la señal `codigo`.

2.b) (0.5 puntos) 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) (0.5 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de un multiplexor 4 a 2 de 2 bits empleando una sentencia **if**.

El circuito multiplexor ha de tener las siguientes señales de entrada: `x0` de 2 bits, `x1` de 2 bits, `x2` de 2 bits, `x3` de 2 bits y una señal de selección `sel` de dos bits. Y ha de tener una señal de salida `smux` de 2 bits.

2.d) (0.5 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito *multiplexor* 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.c. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas.

2.e) (4 puntos) Escriba en VHDL la **architecture** que describe la estructura de un circuito codificador de prioridad de 16 a 4, empleando para ello únicamente codificadores de prioridad 4 a 2 iguales a los diseñados en el Apartado 2.a y el multiplexor diseñado en el Apartado 2.c. Dibuje el diagrama circuital correspondiente a dicho diseño. La tabla de verdad y la **entity** del circuito codificador de prioridad de 16 a 4 se muestran a continuación.

r(15:0)	codigo(3:0)	activo
0000000000000000	0000	'0'
0000000000000001	0000	'1'
000000000000001 -	0001	'1'
00000000000001 --	0010	'1'
0000000000001 ---	0011	'1'
000000000001 ----	0100	'1'
00000000001 -----	0101	'1'
0000000001 ------	0110	'1'
000000001 -------	0111	'1'
00000001 --------	1000	'1'
0000001 ---------	1001	'1'
000001 ----------	1010	'1'
00001 -----------	1011	'1'
0001 ------------	1100	'1'
001 -------------	1101	'1'
01 --------------	1110	'1'
1 ---------------	1111	'1'

```
entity codificadorP16a4 is
  port ( codigo: out std_logic_vector(3 downto 0);
         activo: out std_logic;
         r : in std_logic_vector(15 downto 0) );
end entity codificadorP16a4;
```

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

Solución al Ejercicio 2

El código VHDL de la **entity** y la **architecture** del codificador de prioridad 4 a 2 se muestra en Código VHDL 1.7. El banco de pruebas del codificador de prioridad 4 a 2 se muestra en Código VHDL 1.8–1.9. El cronograma obtenido al simular el banco de pruebas se muestra en la Figura 1.4.

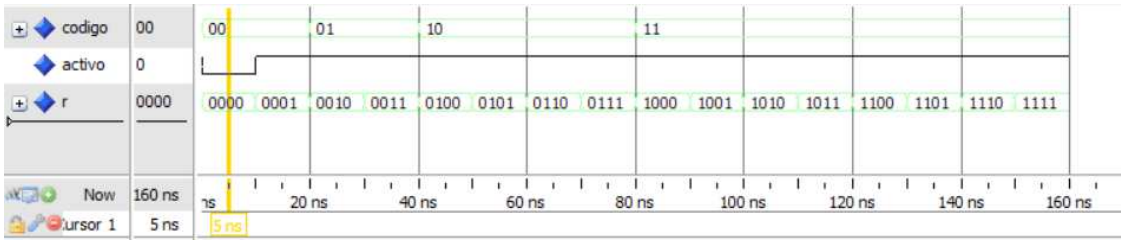

Figura 1.4: Cronograma del Apartado 2.b.

```
----------------------------------------
-- Codificador 4:2 con prioridad
library IEEE;
use IEEE.std_logic_1164.all;

entity codificadorP4a2 is
    port ( activo : out std_logic; -- '1' si hay entrada
           codigo : out std_logic_vector(1 downto 0);
           r : in  std_logic_vector(3 downto 0) );
end entity codificadorP4a2;

architecture codificadorP4a2 of codificadorP4a2 is
begin
   process (r) is -- Activado cuando cambia alguna entrada
   begin
    if ( r="0000" ) then
      activo <= '0';  -- Indica que la salida es válida
    else
      activo <= '1';  -- Salida no válida, puesto que no hay entrada
    end if;
    case r is
      when "0000"      =>
         codigo <= "00";
      when "0001"      =>
         codigo <= "00";
      when "0010"|"0011"      =>
         codigo <= "01";
      when "0100"|"0101"|"0110"|"0111" =>
         codigo <= "10";
      when "1000" |"1001"|"1010"| "1011"|"1100"|"1101"|"1110"|"1111"=>
         codigo <= "11";
      when others =>
         codigo <= "00";
     end case;
   end process;
end architecture codificadorP4a2;
----------------------------------------
```

Código VHDL 1.7: 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 r: std_logic_vector (3 downto 0);

    component codificadorP4a2 is
    port ( activo : out std_logic; -- '1' si hay entrada
           codigo : out std_logic_vector(1 downto 0);
           r : in   std_logic_vector(3 downto 0) );
    end component codificadorP4a2;
  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;
      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);
```

Código VHDL 1.8: Solución al Apartado 2.b: banco de pruebas del codificador de prioridad 4 a 2.

```
        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
          (activo,codigo, r);

    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
        r <= std_logic_vector(to_unsigned(i,4));
        wait for DELAY;
        check_salida(r,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 1.9: Continuación del banco de pruebas del codificador de prioridad 4 a 2.

El código VHDL de la **entity** y la **architecture** del multiplexor 4 a 2 de 2 bits se muestra en Código VHDL 1.10.

El banco de pruebas de este multiplexor está en Código VHDL 1.11. El cronograma obtenido al simular dicho banco de pruebas se puede ver en la Figura 1.5.

```
----------------------------------------
-- Multiplexor 4:2 library IEEE;
library IEEE;
use IEEE.std_logic_1164.all;

entity mux4a2 is
    port ( smux : out std_logic_vector(1 downto 0);
           sel : in std_logic_vector(1 downto 0);
           x0 : in   std_logic_vector(1 downto 0);
           x1 : in   std_logic_vector(1 downto 0);
           x2 : in   std_logic_vector(1 downto 0);
           x3 : in   std_logic_vector(1 downto 0) );
end entity mux4a2;

architecture mux4a2 of mux4a2 is
begin
   process (x0, x1, x2, x3, sel) is
   begin
    if ( sel ="00" ) then
       smux <= x0;
    elsif (sel = "01") then
       smux <= x1;
    elsif (sel = "10") then
       smux <= x2;
    else
       smux <= x3;
    end if;
   end process;
end architecture mux4a2;
----------------------------------------
```

Código VHDL 1.10: Solución al Apartado 2.c: Diseño del multiplexor 4 a 2 de 2 bits.

```
  ------------------------------------------------------
-- Banco de pruebas del codificador de prioridad 4 a 2
-- 10240 ns finaliza la simulacion
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_mux4a2 is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_mux4a2;

architecture bp_mux4a2 of bp_mux4a2 is
    signal smux, sel : std_logic_vector(1 downto 0);
    signal x0, x1, x2, x3 : std_logic_vector (1 downto 0);
    component mux4a2 is
    port (smux : out std_logic_vector(1 downto 0);
             sel : in std_logic_vector(1 downto 0);
             x0 : in   std_logic_vector(1 downto 0);
             x1 : in   std_logic_vector(1 downto 0);
             x2 : in   std_logic_vector(1 downto 0);
             x3 : in   std_logic_vector(1 downto 0));
   end component mux4a2;
   procedure check_salida
       ( smux     : in std_logic_vector(1 downto 0);
         sel        : in std_logic_vector(1 downto 0);
         x0         : in std_logic_vector(1 downto 0);
         x1         : in std_logic_vector(1 downto 0);
         x2         : in std_logic_vector(1 downto 0);
         x3         : in std_logic_vector(1 downto 0);
         error_count: inout integer ) is
         variable smux_esp: std_logic_vector(1 downto 0);
   begin
    case sel is
       when "00"       =>
          smux_esp := x0;
       when "01"       =>
          smux_esp := x1;
       when "10"       =>
          smux_esp := x2;
       when others =>
          smux_esp := x3;
    end case;
```

Código VHDL 1.11: Solución al Apartado 2.d: banco de pruebas del multiplexor 4 a 2 de 2 bits.

```
      assert( smux = smux_esp)
        report "ERROR. Entrada: " & std_logic'image(sel(1))
                  & std_logic'image(sel(0)) & std_logic'image(x0(1)) &
                  std_logic'image(x0(0)) &   std_logic'image(x1(1)) &
                  std_logic'image(x1(0)) &   std_logic'image(x2(1)) &
                  std_logic'image(x2(0)) &      std_logic'image(x3(1)) &
                  std_logic'image(x3(0)) &
                  ",resultado esperado: " &
                  std_logic'image(smux_esp(1)) & std_logic'image(smux_esp(0)) &

                  ",resultado actual: " &
                   std_logic'image(smux(1)) & std_logic'image(smux(0)) &
                  " en el instante " &
                  time'image(now);
    if (smux /= smux_esp) then
      error_count := error_count + 1;
    end if;
   end procedure check_salida;
   -- Fin de la definición del procedure
begin
    UUT : component mux4a2 port map
          (smux,sel, x0, x1, x2, x3);

vec_test : process is
      variable error_count : integer := 0;
      variable r: std_logic_vector(9 downto 0);
    begin
      report "Comienza la simulación";
      -- Generar todos los posibles valores de entrada
      for i in 0 to 2**10-1 loop
        r := std_logic_vector(to_unsigned(i,10));
        sel <= r(1 downto 0);
        x0 <= r(3 downto 2);
        x1 <= r(5 downto 4);
        x2 <= r(7 downto 6);
        x3 <= r(9 downto 8);
        wait for DELAY;
        check_salida(smux, sel, x0, x1, x2, x3, 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_mux4a2;
-------------------------------------------------
```

Código VHDL 1.12: Solución al Apartado 2.d: continuación del banco de pruebas del multiplexor 4 a 2.

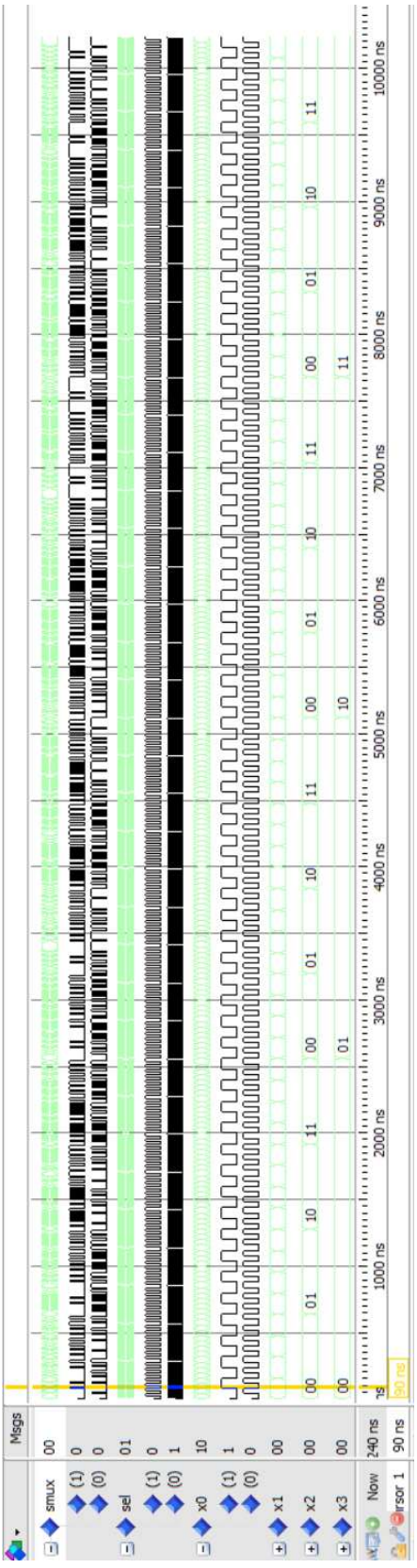

Figura 1.5: Cronograma del Apartado 2.d.

El diagrama circuital correspondiente al diseño estructural del codificador de prioridad 16 a 4 se muestran en la Figura 1.6. El diseño estructural del codificador de prioridad 16 a 4 se muestra en Código VHDL 1.13 y su banco de pruebas en 1.14–1.16. El cronograma obtenido tras simular el banco de pruebas se muestra en 1.7.

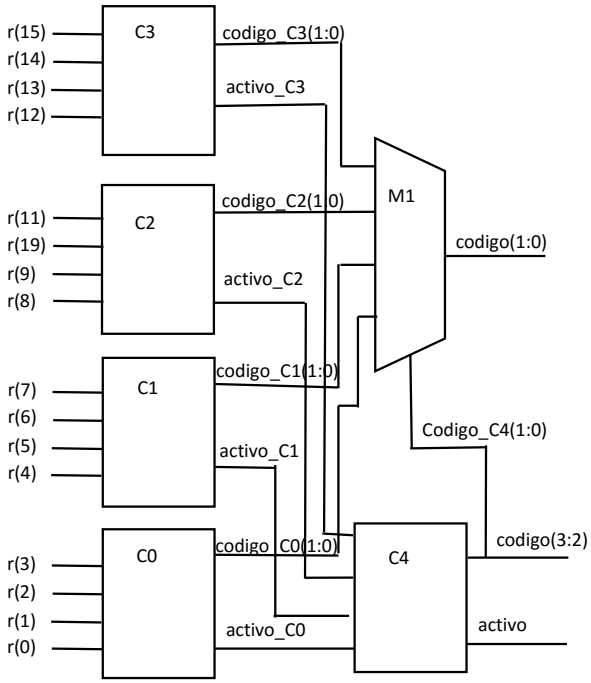

Figura 1.6: Diagrama circuital del codificador de prioridad 16 a 4.

```
  -- Codificador 16:4 con prioridad
library IEEE;
use IEEE.std_logic_1164.all;

entity codificadorP16a4 is
    port ( activo : out std_logic; -- '1' si hay entrada
           codigo : out std_logic_vector(3 downto 0);
           r : in  std_logic_vector(15 downto 0) );
end entity codificadorP16a4;

architecture codificadorP16a4 of codificadorP16a4 is
    signal codigo_C4, codigo_C3, codigo_C2, codigo_C1, codigo_C0:
std_logic_vector(1 downto 0);
    signal activo_C3, activo_C2, activo_C1, activo_C0: std_logic;
    signal r_C4: std_logic_vector(3 downto 0);
    component mux4a2 is
     port ( smux : out std_logic_vector(1 downto 0);
           sel : in std_logic_vector(1 downto 0);
           x0 : in  std_logic_vector(1 downto 0);
           x1 : in  std_logic_vector(1 downto 0);
           x2 : in  std_logic_vector(1 downto 0);
           x3 : in  std_logic_vector(1 downto 0) );
    end component mux4a2;
    component codificadorP4a2 is
    port ( activo : out std_logic; -- '1' si hay entrada
           codigo : out std_logic_vector(1 downto 0);
           r : in  std_logic_vector(3 downto 0) );
    end component codificadorP4a2;
begin
-- Instanciación y conexión de los componentes
  C3 : component codificadorP4a2 port map (activo_C3, codigo_C3, r(15
downto 12));
  C2 : component codificadorP4a2 port map (activo_C2, codigo_C2, r(11
downto 8));
  C1 : component codificadorP4a2 port map (activo_C1, codigo_C1, r(7
downto 4));
  C0 : component codificadorP4a2 port map (activo_C0, codigo_C0, r(3
downto 0));
  C4 : component codificadorP4a2 port map (activo, codigo_C4, r_C4);
  M1 : component mux4a2 port map (codigo(1 downto 0), codigo_C4,
codigo_C0,
           codigo_C1, codigo_C2, codigo_C3);
  codigo(3 downto 2) <= codigo_C4;
  r_C4(3) <= activo_C3;
  r_C4(2) <= activo_C2;
  r_C4(1) <= activo_C1;
  r_C4(0) <= activo_C0;
end architecture codificadorP16a4;
---------------------------------------
```

Código VHDL 1.13: Solución al Apartado 2.e: Diseño del codificador de prioridad de 16 a 4.

```
  ----------------------------------------------------
-- Banco de pruebas del codificador de prioridad 16 a 4
-- La simulacion finaliza en t = 655360 ns
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_codificadorP16a4 is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_codificadorP16a4;

architecture bp_codificadorP16a4 of bp_codificadorP16a4 is
    signal codigo : std_logic_vector(3 downto 0);
    signal activo   : std_logic;
    signal r: std_logic_vector (15 downto 0);

    component codificadorP16a4 is
    port ( activo : out std_logic; -- '1' si hay entrada
           codigo : out std_logic_vector(3 downto 0);
           r : in   std_logic_vector(15 downto 0) );
    end component codificadorP16a4;
  procedure check_salida
      ( x      : in std_logic_vector(15 downto 0);
        codigo         : in std_logic_vector(3 downto 0);
        activo       : in std_logic;
        error_count: inout integer ) is
        variable codigo_esp: std_logic_vector(3 downto 0);
        variable activo_esp: std_logic;
  begin
       if std_match(x,"0000000000000000") then codigo_esp := "0000";
       elsif std_match(x,"0000000000000001") then codigo_esp := "0000";
       elsif std_match(x,"000000000000001-") then codigo_esp := "0001";
       elsif std_match(x,"00000000000001--")  then  codigo_esp := "0010";
       elsif std_match(x,"0000000000001---")  then codigo_esp := "0011";
       elsif std_match(x,"000000000001----")  then codigo_esp := "0100";
       elsif std_match(x,"00000000001-----") then codigo_esp := "0101";
       elsif std_match(x,"0000000001------") then codigo_esp := "0110";
       elsif std_match(x,"000000001-------") then codigo_esp := "0111";
       elsif std_match(x,"00000001--------") then codigo_esp := "1000";
       elsif std_match(x,"0000001---------") then codigo_esp := "1001";
       elsif std_match(x,"000001----------") then codigo_esp := "1010";
       elsif std_match(x,"00001-----------") then codigo_esp := "1011";
       elsif std_match(x,"0001------------") then codigo_esp := "1100";
       elsif std_match(x,"001-------------") then codigo_esp := "1101";
       elsif std_match(x,"01--------------")  then codigo_esp := "1110";
       elsif std_match(x,"1---------------") then codigo_esp := "1111";
       else codigo_esp := "0000";
     end if;
      if (x = "0000000000000000") then activo_esp := '0';
      else activo_esp:= '1';
      end if;
```

Código VHDL 1.14: Solución al Apartado 2.f: banco de pruebas del codificador de prioridad de 16 a 4.

```
          assert( codigo = codigo_esp)
         report "ERROR. Entrada: " & std_logic'image(x(15))
                  & std_logic'image(x(14)) & std_logic'image(x(13)) &
                  std_logic'image(x(12)) & std_logic'image(x(11)) &
                  std_logic'image(x(10)) & std_logic'image(x(9)) &
                  std_logic'image(x(8)) & 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(3))                                    &
std_logic'image(codigo_esp(2)) &
                  std_logic'image(codigo_esp(1))                                    &
std_logic'image(codigo_esp(0)) &
                  ", resultado actual: " &
                   std_logic'image(codigo(3)) & 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(15))
                  & std_logic'image(x(14)) & std_logic'image(x(13)) &
                  std_logic'image(x(12)) & std_logic'image(x(11)) &
                  std_logic'image(x(10)) & std_logic'image(x(9)) &
                  std_logic'image(x(8)) & 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
```

Código VHDL 1.15: Solución al Apartado 2.f: continuación del banco de pruebas del codificador de prioridad 16 a 4.

```
begin
    UUT : component codificadorP16a4 port map
          (activo,codigo, r);

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 2**16-1 loop
         r <= std_logic_vector(to_unsigned(i,16));
         wait for DELAY;
         check_salida(r,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_codificadorP16a4;
-------------------------------------------------
```

Código VHDL 1.16: Solución al Apartado 2.f: continuación del banco de pruebas del codificador de prioridad 16 a 4.

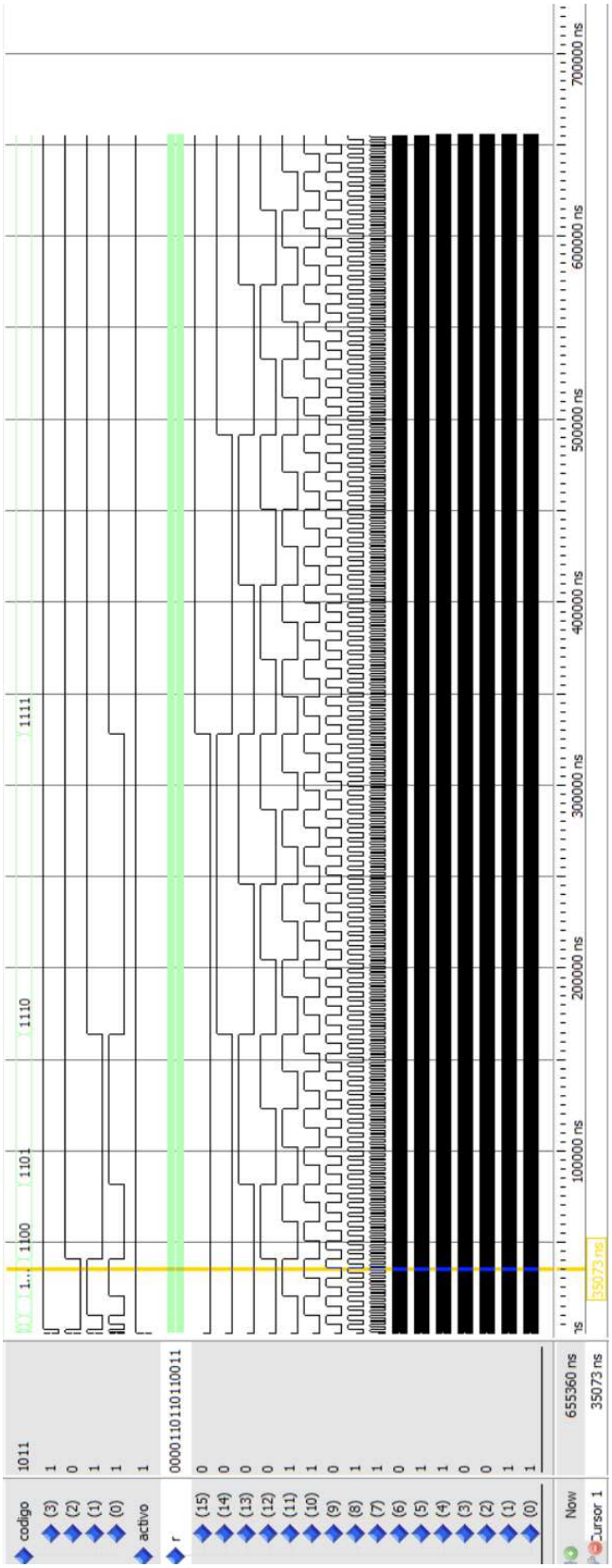

Figura 1.7: Cronograma del Apartado 2.f.