INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2015

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'	'1'
'0'	'0'	'1'	'1'
'0'	'1'	'0'	'0'
'0'	'1'	'1'	'0'
'1'	'0'	'0'	'1'
'1'	'0'	'1'	'0'
'1'	'1'	'0'	'1'
'1'	'1'	'1'	'0'

1.a) (0.5 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 la salida F.

1.b) (0.5 puntos) 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.c) (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.d) (1 punto) Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, la salida del circuito diseñado en el Apartado 1.c. Compruebe mediante inspección visual que el diseño funciona correctamente. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas.

Solución al Ejercicio 1

La función lógica obtenida es la siguiente:

```
F = (not x and not y) or (x and not z)
```

La **entity** del circuito que implementa la función lógica se muestra en Código VHDL 1.1.

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

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

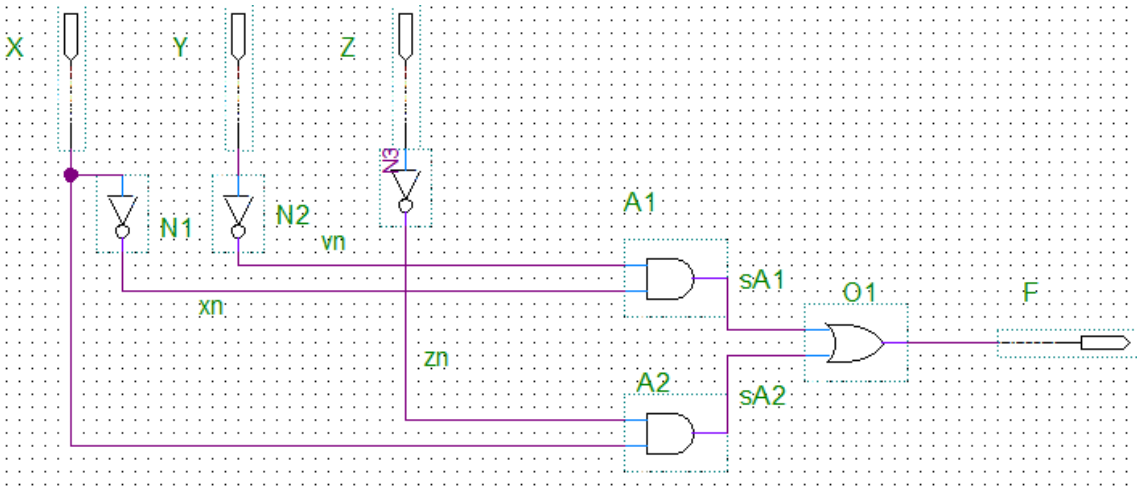

Figura 1.1: Solución al Apartado 1.b: diagrama al nivel de puertas lógicas.

El código VHDL de la **entity** y la **architecture** de estas tres puertas lógicas se muestra en Código VHDL 1.2, 1.3 y 1.4, respectivamente. El Código VHDL 1.5 muestra la **architecture** del circuito describiendo 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 1.2: 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 1.3: 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 1.4: Puerta NOT lógica.

```
-----------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcF is
   signal xn, yn, zn: std_logic;
   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 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);
   N2 : component not1 port map (yn, y);
   N3 : component not1 port map (zn, z);
   A1 : component and2 port map (sA1, xn, yn);
   A2 : component and2 port map (sA2, x, zn);
   O1 : component or2 port map (F, sA1, sA2);
end architecture circuito_Estruc;
---------------------------------
```

Código VHDL 1.5: Solución al Apartado 1.c: **architecture** del circuito describiendo su estructura.

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.6. El cronograma obtenido al simular el banco de pruebas es mostrado en la Figura 1.2.

```
--------------------------------------------------
-- 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.d: banco de pruebas del circuito.

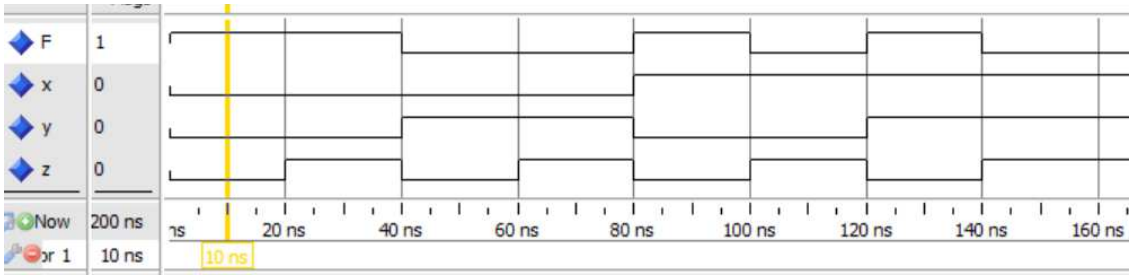

Figura 1.2: Cronograma del Apartado 1.d.

EJERCICIO 2

2.a) (0.5 puntos) Escriba en VHDL la **architecture** que describe el comportamiento de un circuito multiplexor de dos señales de 1 bit (`a` y `b`) empleando para ello un bloque **process** y una sentencia **if**. El circuito ha de tener la siguiente **entity**.

```
entity mux2a1 is
   port ( y  : out std_logic;
          a  : in  std_logic;
          b  : in  std_logic;
          s  : in std_logic );
end entity mux2a1;
```

2.b) (1.5 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.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 combinacional desplazador de 4 bits, empleando para ello únicamente multiplexores 2 a 1 iguales al diseñado en el Apartado 2.a. La tabla de operación y la **entity** del circuito desplazador se muestran a continuación.

Tabla 1: Operaciones del desplazador.

s(1)s(0)	**Operación**
00	Rota un bit a la derecha
01	Rota un bit a la izquierda
10	Desplazada un bit a la derecha, introduciendo un '0' en el bit más significativo
11	Desplaza un bit a la izquierda, introduciendo un '0' en el bit menos significativo

```
entity desplazador is
   port ( O  : out std_logic_vector(3 downto 0);
          I  : in  std_logic_vector(3 downto 0);
          s  : in  std_logic_vector(1 downto 0) );
end entity desplazador;
```

2.d) (2 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito *desplazador* 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

La **architecture** describiendo el comportamiento del circuito multiplexor se muestra en Código VHDL 1.7. El banco de pruebas se muestra en Código VHDL 1.8. El cronograma obtenido al simular el banco de pruebas se muestra en la Figura 1.3.

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

entity mux2a1 is
     port ( y   : out std_logic;
                a   : in    std_logic;--1
                b   : in    std_logic;--0
                s   : in std_logic );
end entity mux2a1;

architecture mux2a1 of mux2a1 is
begin
process(a, b, s)
begin
  if (s='0') then
     y <= b;
  else
     y <= a;
  end if;
end process;
end architecture mux2a1;
----------------------------------
```

Código VHDL 1.7: Solución al Apartado 2.a: **architecture** del circuito mux 2:1.

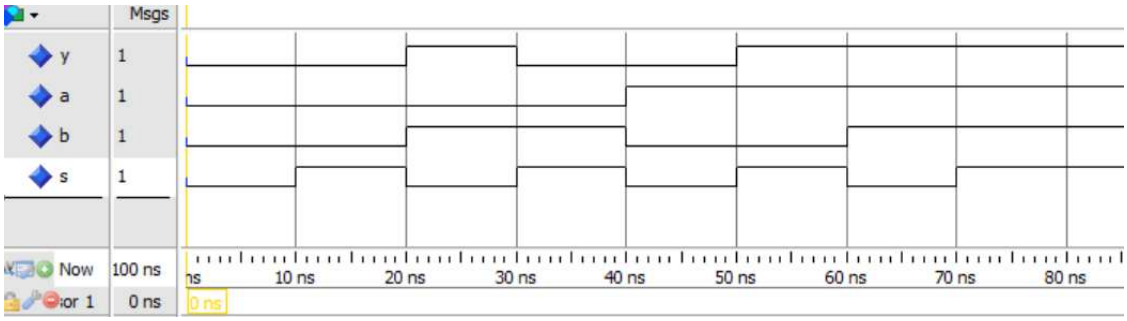

Figura 1.3: Cronograma del Apartado 2.b.

```
-- Banco de pruebas del mux 2:1
-------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_mux2a1 is
    constant DELAY      : time      := 10 ns; -- Retardo usado en
                                              el test
end entity bp_mux2a1;

architecture bp_mux2a1 of bp_mux2a1 is
    signal y : std_logic;
    signal a, b, s   : std_logic;
    component mux2a1 is
      port (   y      : out std_logic;
               a, b, s  : in std_logic );
      end component mux2a1;
begin
    UUT : component mux2a1 port map (y, a, b, s);
    vec_test : process is
       variable valor  : unsigned (2 downto 0);
       variable y_e :std_logic;
       variable num_errores: integer := 0;
    begin
       -- Generar todos los posibles valores de entrada
       for i in 0 to 7 loop
          valor := to_unsigned(i,3);
          a <= std_logic(valor(2));
          b <= std_logic(valor(1));
          s <= std_logic(valor(0));
          wait for DELAY;
          if (s='0') then
             y_e := b;
          else
             y_e := a;
          end if;
          if ( y_e /= y) then
             report "Error en el instante "& time'image(now);
             num_errores := num_errores+1;
          end if;
       end loop;
       report "Test completo. Hay "& integer'image(num_errores
                                                    ) & "
                                                    errores.";
       wait;      -- Final de la simulación
   end process vec_test;
end architecture bp_mux2a1;
-------------------------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.b: banco de pruebas del circuito mux 2:1.

El diagrama de la estructura del circuito desplazador compuesto usando los multiplexores 2:1 se muestra en la Figura 1.4. La **architecture** del circuito desplazador se muestra en Código VHDL 1.9. El banco de pruebas se muestra en Código VHDL 1.10–1.11. El cronograma obtenido al simular el banco de pruebas se muestra en la Figura 1.5.

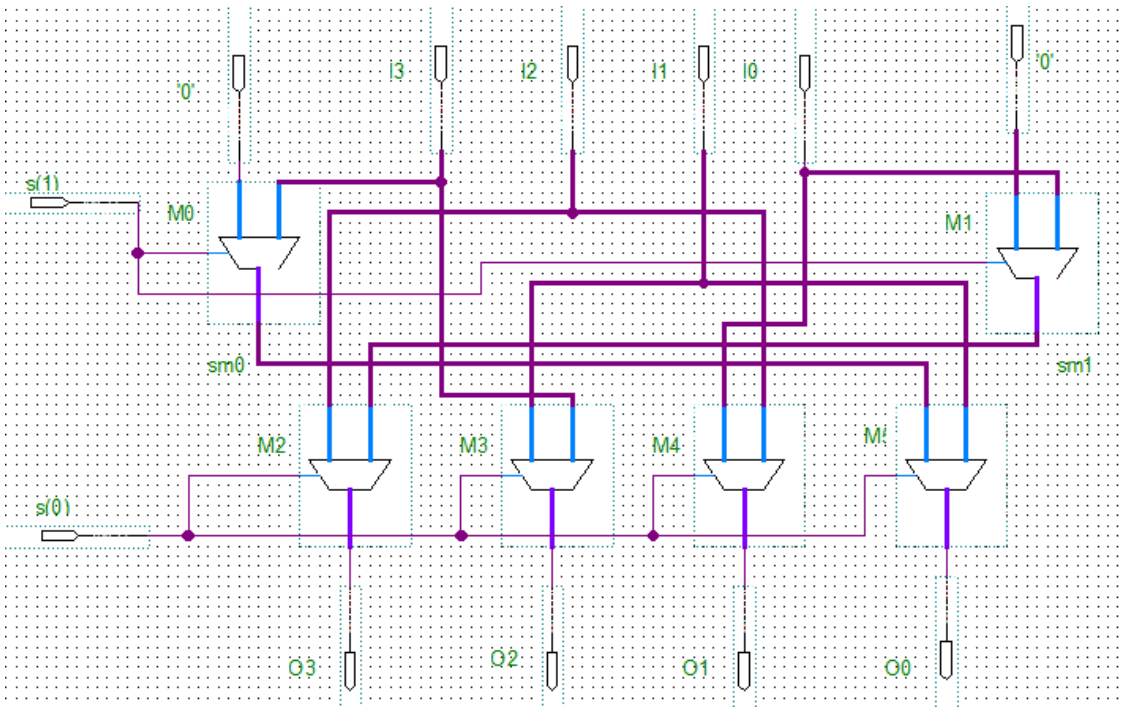

Figura 1.4: Diagrama estructural del circuito desplazador

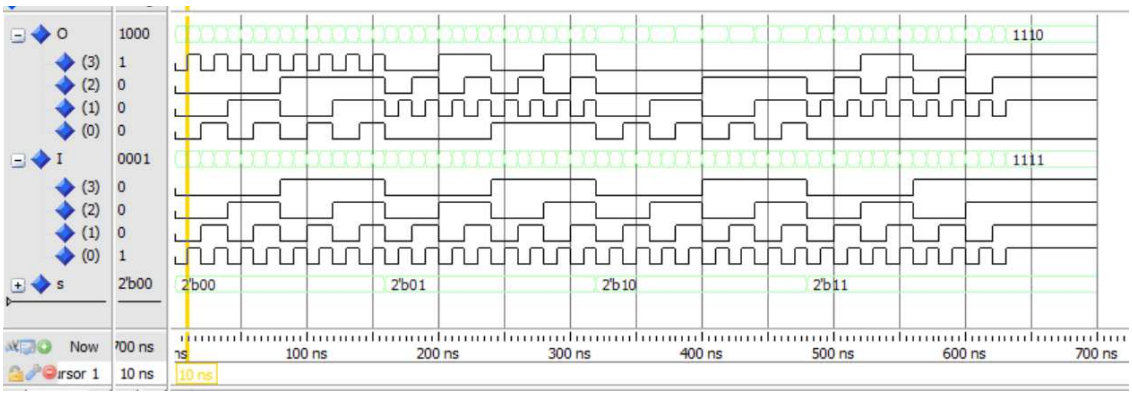

Figura 1.5: Cronograma del Apartado 2.d.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of desplazador is
   signal sm0, sm1: std_logic;
-- Declaración componente
   component mux2a1 is
     port ( y   : out std_logic;
            a   : in    std_logic;--1
            b   : in    std_logic;--0
            s   : in std_logic );
   end component mux2a1;
begin
-- Instanciación y conexión de los componentes
   M0 : component mux2a1 port map (sm0, '0', I(3), s(1));
   M1 : component mux2a1 port map (sm1, '0', I(0), s(1));
   M2 : component mux2a1 port map (O(3), I(2), sm1, s(0));
   M3 : component mux2a1 port map (O(2), I(1), I(3), s(0));
   M4 : component mux2a1 port map (O(1), I(0), I(2), s(0));
   M5 : component mux2a1 port map (O(0), sm0, I(1), s(0));
end architecture circuito_Estruc;
-----------------------------------
```

Código VHDL 1.9: Solución al Apartado 2.c: **architecture** del circuito desplazador.

```
-------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_desplazador is
    constant DELAY      : time      := 10 ns; -- Retardo usado en
                                              el test
end entity bp_desplazador;
architecture bp_desplazador of bp_desplazador is
    signal O, I  : std_logic_vector(3  downto 0);
    signal s     : std_logic_vector(1  downto 0);

    procedure error_check( I: in std_logic_vector(3  downto 0);
                           s: in std_logic_vector(1  downto 0);
                           actual: in std_logic_vector (3
                                                        downto
                                                        0);
                           num_errores: inout integer) is
     variable esperado: std_logic_vector(3  downto 0);
    begin
     if (s="00") then
        esperado := I(0)&I(3)&I(2)&I(1);
     elsif (s="01") then
        esperado := I(2)&I(1)&I(0)&I(3);
     elsif (s="10") then
        esperado := '0'&I(3)&I(2)&I(1);
     else
        esperado := I(2)&I(1)&I(0)&'0';
     end if;
     if (esperado /= actual) then
        report "Error en el instante "& time'image(now);
        num_errores := num_errores+1;
     end if;
    end procedure error_check;
    component desplazador is
      port ( O   : out std_logic_vector(3  downto 0);
             I   : in  std_logic_vector(3  downto 0);
             s   : in  std_logic_vector(1  downto 0) );
    end component desplazador;
```

Código VHDL 1.10: Solución al Apartado 2.d: banco de pruebas del circuito desplazador.

```
begin
    UUT : component desplazador  port map
          (O, I, s);

vec_test : process is
      variable valor   : unsigned (5  downto 0);
      variable O_e : std_logic_vector(3  downto 0);
      variable num_errores: integer := 0;
    begin
      -- Generar todos los posibles valores de entrada
      for j  in 0  to 2**6-1  loop
        valor := to_unsigned(j,6);
        s  <= std_logic(valor(5))&std_logic(valor(4));
        I(3) <= std_logic(valor(3));
        I(2) <= std_logic(valor(2));
        I(1) <= std_logic(valor(1));
        I(0) <= std_logic(valor(0));
        wait for DELAY;
        error_check(I, s, O, num_errores);
      end loop;
      report "Test completo. Hay "& integer'image(num_errores
                                                  ) & "
                                                  errores.";
      wait;      -- Final de la simulación
   end process vec_test;
end architecture bp_desplazador;
-------------------------------------------------
```

Código VHDL 1.11: Solución al Apartado 2.d: continuación del banco de pruebas del circuito desplazador.