INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2012

EJERCICIO 1

1.a) (0.5 puntos) Escriba en VHDL la **entity** de un circuito multiplexor de dos señales de 1 bit. Como se muestra en la Figura 1.1, el circuito debe tener dos entradas de datos de un bit, llamadas `x0` y `x1`, una entrada de selección de un bit, llamada `s`, y una salida de un bit, llamada `y`.

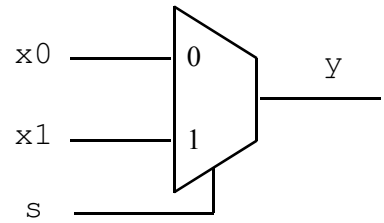

Figura 1.1: Multiplexor de 2 entradas de 1 bit.

1.b) (1 punto) Escriba la tabla de la verdad del circuito multiplexor de dos señales de 1 bit. A partir de dicha tabla de la verdad, obtenga la función lógica que describe la salida (`y`) en función de las entradas (`x0`, `x1`, `s`). A continuación, escriba en VHDL una **architecture** que describa el *comportamiento* de un circuito que implemente dicha función lógica.

1.c) (0.5 puntos) Dibuje el diagrama al nivel de puertas lógicas de un circuito que implemente esta función. 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, la salida de los circuitos diseñados en los Apartados 1.b y 1.d. Compruebe mediante inspección visual que los dos diseños funcionan correctamente.

Solución al Ejercicio 1

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

entity MUX_2a1_1bit is
   port ( y          : out std_logic;
          x0, x1     : in  std_logic;
          s          : in  std_logic );
end entity MUX_2a1_1bit;
-------------------------------------
```

Código VHDL 1.1: Apartado 1.a: **entity** del circuito multiplexor de dos señales de 1 bit.

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

architecture MUX_2a1_1bit_funcLog of MUX_2a1_1bit is
begin
   y <= (s and x1) or (not s and x0);
end architecture MUX_2a1_1bit_funcLog;
-------------------------------------
```

Código VHDL 1.2: Apartado 1.b: **architecture** que describe el comportamiento de la función lógica.

```
-------------------------------------
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;
-------------------------------------

-------------------------------------
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;
-------------------------------------

-------------------------------------
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 1.3: Apartado 1.c: diseño de las puertas lógicas.

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

architecture MUX_2a1_1bit_struct of MUX_2a1_1bit is
   signal not_s, s_x1, x0_nots: 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 not1 port map (not_s, s);
   g1 : component and2 port map (s_x1, s, x1);
   g2 : component and2 port map (x0_nots, x0, not_s);
   g3 : component or2 port map (y, s_x1, x0_nots);
end architecture MUX_2a1_1bit_struct;
-----------------------------------
```

Código VHDL 1.4: Apartado 1.d: **architecture** que describe la estructura del circuito.

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

entity bp_MUX is
end entity bp_MUX;

architecture bp_MUX_visual of bp_MUX is
   signal y            : std_logic;   -- Salidas UUT
   signal x0, x1, s    : std_logic;   -- Entradas UUT

   component MUX_2a1_1bit is
        port ( y          : out std_logic;
               x0, x1     : in  std_logic;
               s          : in  std_logic );
   end component MUX_2a1_1bit;

begin
   -- Instanciar y conectar UUT
   uut : component MUX_2a1_1bit port map
           ( y => y,
             x0 => x0, x1 => x1, s => s );
   gen_vec_test : process
      variable test_in : unsigned (2 downto 0); -- Vector de test
   begin
      test_in := B"000";
      for count in 0 to 7 loop
         s      <= 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_MUX_visual;
-------------------------------------
```

Código VHDL 1.5: Apartado 1.e: banco de pruebas.

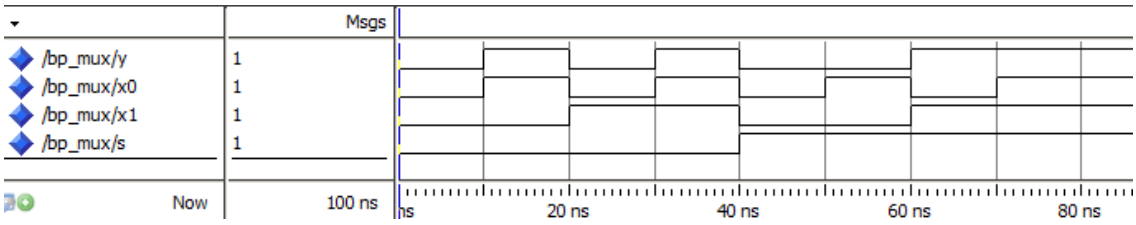

Figura 1.2: Simulación del banco de pruebas.

EJERCICIO 2

En la Figura 1.3 se muestra la estructura de un circuito combinacional desplazador denominado *barrel shifter*. Obsérvese que está compuesto por la conexión de tres etapas, cada una de las cuales está compuesta por ocho multiplexores (MUX) dispuestos en una misma columna. Cada uno de estos multiplexores es como el diseñado en el Ejercicio 1.

En la primera etapa, que es la columna de multiplexores situada más a la izquierda, una de las entradas a uno de los multiplexores está conectada a `'0'`. En la segunda etapa están conectadas a `'0'` una de las entradas de dos multiplexores y en la tercera etapa una de las entradas de cuatro multiplexores.

Las entradas de selección de todos los multiplexores de la primera etapa están conectadas entre sí y conectadas a la entrada `s(0)`. Igualmente, las entradas de selección de todos los multiplexores de la segunda etapa están conectadas entre sí y conectadas a la entrada `s(1)`. Lo mismo para las entradas de selección de los multiplexores de la tercera etapa, que están conectadas a `s(2)`.

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

```
entity barrelShifter is
   port ( y  : out std_logic_vector(7 downto 0);
          x  : in  std_logic_vector(7 downto 0);
          s  : in  std_logic_vector(2 downto 0) );
end entity barrelShifter;
```

2.a) (2 puntos) Diseñe en VHDL la **architecture** que describe el comportamiento del circuito combinacional desplazador *barrel shifter*.

2.b) (2 puntos) Diseñe en VHDL la **architecture** que describe la estructura del circuito combinacional desplazador *barrel shifter*, mediante la conexión de los multiplexores como el diseñado al resolver el Ejercicio 1 (puede emplear indistintamente el diseño realizado en el Apartado 1.b ó 1.d).

2.c) (2 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito *barrel shifter*. 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 los diseños realizados al contestar a los Apartados 2.a y 2.b.

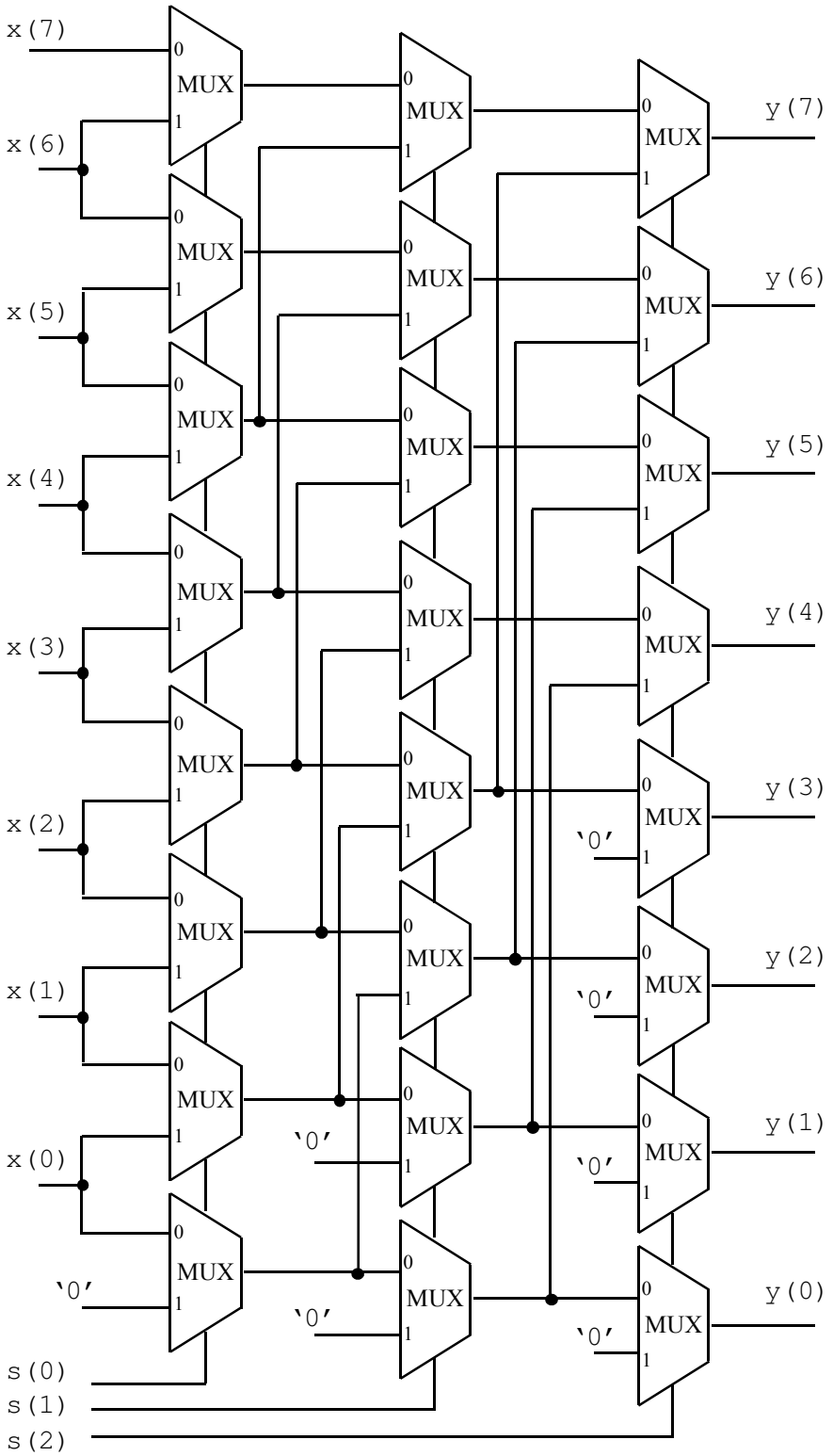

Figura 1.3: Diagrama del circuito desplazador denominado "barrel shifter".

Solución al Ejercicio 2

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

architecture barrelShifter_comp of barrelShifter is
begin
   process (x,s)
      variable e1 : std_logic_vector(7 downto 0); -- Salida etapa 1
      variable e2 : std_logic_vector(7 downto 0); -- Salida etapa 2
   begin
      --- Primera etapa ---
      if ( s(0) = '0') then
         e1 := x;
      else
         e1(0) := '0';
         for i in 1 to 7 loop
            e1(i) := x(i-1);
         end loop;
      end if;
      --- Segunda etapa ---
      if ( s(1) = '0') then
         e2 := e1;
      else
         e2(0) := '0';
         e2(1) := '0';
         for i in 2 to 7 loop
            e2(i) := e1(i-2);
         end loop;
      end if;
      --- Tercera etapa ---
      if ( s(2) = '0') then
         y <= e2;
      else
         y(0) <= '0';
         y(1) <= '0';
         y(2) <= '0';
         y(3) <= '0';
         for i in 4 to 7 loop
            y(i) <= e2(i-4);
         end loop;
      end if;
   end process;
end architecture barrelShifter_comp;
-------------------------------------
```

Código VHDL 1.6: Apartado 2.a: **architecture** que describe el comportamiento del circuito *barrel shifter*.

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

architecture barrelShifter_estruc of barrelShifter is
   signal e1 : std_logic_vector(7 downto 0); -- Salida etapa 1
   signal e2 : std_logic_vector(7 downto 0); -- Salida etapa 2
   -- Declaración de la clase de los componentes
   component MUX_2a1_1bit is
        port ( y          : out std_logic;
               x0, x1     : in  std_logic;
               s          : in  std_logic );
   end component MUX_2a1_1bit;
begin
   -- Primera etapa --
   etapa1: for i in 7 downto 0 generate
   begin
        e1_MUX_zero: if i=0 generate
             mux_1_0: MUX_2a1_1bit port map (e1(0),x(0),'0',s(0));
        end generate e1_MUX_zero;
        e1_MUX_no_zero: if i /= 0 generate
             mux_1 : MUX_2a1_1bit port map (e1(i),x(i),x(i-1),s(0));
        end generate e1_MUX_no_zero;
   end generate etapa1;
   -- Segunda etapa --
   etapa2: for i in 7 downto 0 generate
   begin
        e2_MUX_zero: if i<2 generate
             mux_2_0: MUX_2a1_1bit port map (e2(i),e1(i),'0',s(1));
        end generate e2_MUX_zero;
        e2_MUX_no_zero: if i>1 generate
             mux_2 : MUX_2a1_1bit port map (e2(i),e1(i),e1(i-2),s(1));
        end generate e2_MUX_no_zero;
   end generate etapa2;
   -- Tercera etapa --
   etapa3: for i in 7 downto 0 generate
   begin
        e3_MUX_zero: if i<4 generate
             mux_3_0: MUX_2a1_1bit port map (y(i),e2(i),'0',s(2));
        end generate e3_MUX_zero;
        e3_MUX_no_zero: if i>3 generate
             mux_3 : MUX_2a1_1bit port map (y(i),e2(i),e2(i-4),s(2));
        end generate e3_MUX_no_zero;
   end generate etapa3;
end architecture barrelShifter_estruc;
-----------------------------------
```

Código VHDL 1.7: Apartado 2.b: **architecture** que describe la estructura del circuito *barrel shifter*.

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

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

architecture bp_barrelShifter of bp_barrelShifter is
   signal y : std_logic_vector(7 downto 0); -- Salida UUT
   signal x : std_logic_vector(7 downto 0); -- Entrada UUT
   signal s : std_logic_vector(2 downto 0); -- Entrada UUT

   component barrelShifter is
      port ( y : out std_logic_vector(7 downto 0);
             x : in  std_logic_vector(7 downto 0);
             s : in  std_logic_vector(2 downto 0) );
   end component barrelShifter;
```

Código VHDL 1.8: Apartado 2.c: banco de pruebas para el circuito *barrel shifter* (1/2).

```
begin
   -- Instanciar y conectar UUT
   uut : component barrelShifter port map
         ( y => y, x => x, s => s );
   gen_vec_test : process
      variable esperado_y : std_logic_vector(7 downto 0);
      variable error_count: integer      := 0;   -- Num. total errores
   begin
    for i in 0 to (MAX_COMB-1) loop
        x  <= std_logic_vector(TO_UNSIGNED(i,8));
        for count in 0 to 7 loop
           s <= std_logic_vector(TO_UNSIGNED(count,3));
           esperado_y := std_logic_vector(TO_UNSIGNED(i,8) sll count);
           wait for 10 ns;
           if ( esperado_y /= y ) then
              report "ERROR en la salida valida. Valor esperado: "   &
                     std_logic'image(esperado_y(7))  &
                     std_logic'image(esperado_y(6))  &
                     std_logic'image(esperado_y(5))  &
                     std_logic'image(esperado_y(4))  &
                     std_logic'image(esperado_y(3))  &
                     std_logic'image(esperado_y(2))  &
                     std_logic'image(esperado_y(1))  &
                     std_logic'image(esperado_y(0))  &
                     ", valor actual: "                &
                     std_logic'image(y(7))           &
                     std_logic'image(y(6))           &
                     std_logic'image(y(5))           &
                     std_logic'image(y(4))           &
                     std_logic'image(y(3))           &
                     std_logic'image(y(2))           &
                     std_logic'image(y(1))           &
                     std_logic'image(y(0))           &
                     " en el instante: "               &
                     time'image(now);
              error_count := error_count + 1;
           end if;
        end loop;
      end loop;
      -- Informe del número total de errores
      report   "Hay "                 &
                integer'image(error_count) &
                " errores.";
      wait;     -- Final de la simulación
   end process gen_vec_test;
end architecture bp_barrelShifter;
-------------------------------------
```

Código VHDL 1.9: Apartado 2.c: banco de pruebas para el circuito *barrel shifter* (2/2).