INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2013

EJERCICIO 1

En la Figura 1.1 se muestra el símbolo lógico de un circuito digital cuya función es encender una luz de aviso de un coche. Este circuito enciende la luz de aviso cuando se cumplen simultáneamente las tres condiciones siguientes: la llave del coche está puesta en el contacto, existe un pasajero en el asiento del copiloto y no está abrochado el cinturón del copiloto.

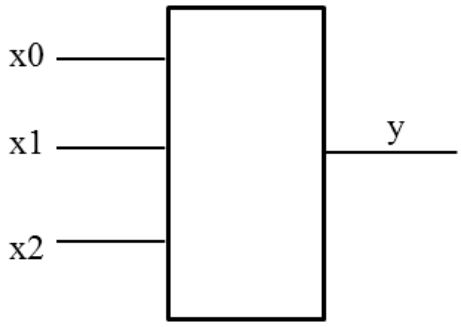


Figura 1.1: Entradas y salida del circuito de luz de aviso.

La señal `x0` está a 1 sólo si la llave del coche está en el contacto, `x1` está a 1 sólo si existe un pasajero en el asiento del copiloto y `x3` está a 1 sólo si el cinturón del asiento del copiloto está abrochado. La señal de salida `y` está a 1 sólo cuando la luz de aviso está encendida.

1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito digital de alarma empleando el mismo nombre para las señales que el mostrado en la Figura 1.1.

1.b) (1 punto) Escriba la tabla de la verdad del circuito digital de alarma. A partir de dicha tabla de la verdad, obtenga la función lógica que describe

la salida (`y`) en función de las entradas (`x0`, `x1`, `x2`). 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 de dos entradas y 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

La **entity** del circuito de alarma se muestra en Código VHDL 1.1.

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

entity alarma is
   port (y: out std_logic;
         x0, x1, x2: in std_logic);
end entity alarma;
-------------------------------------
```

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

La tabla de verdad del circuito es la siguiente:

x2	x1	x0	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

El Código VHDL 1.2 muestra la **architecture** del circuito describiendo su comportamiento.

```
-------------------------------------
architecture comportamiento of alarma is
begin
   y <= x0 and x1 and not x2;
end architecture comportamiento;
-------------------------------------
```

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

La Figura 1.2 muestra el diagrama del circuito empleando dos puertas AND y un inversor.

El código VHDL de la **entity** y la **architecture** de estas dos puertas lógicas se muestra en Código VHDL 1.3 y 1.4, respectivamene.

El Código VHDL 1.5 muestra la **architecture** del circuito describiendo estructura.

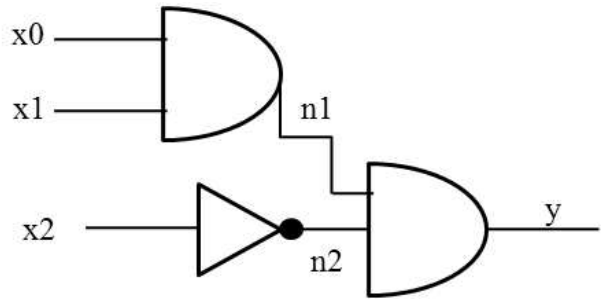


Figura 1.2: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.6.

```
--------------------------------------
architecture estructura of alarma is
   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;
   signal n1, n2: std_logic;
begin
   and2_1: and2 port map (n1, x0, x1);
   inv_1: not1 port map(n2, x2);
   and2_2: and2 port map(y, n1, n2);
end architecture estructura;
--------------------------------------
```

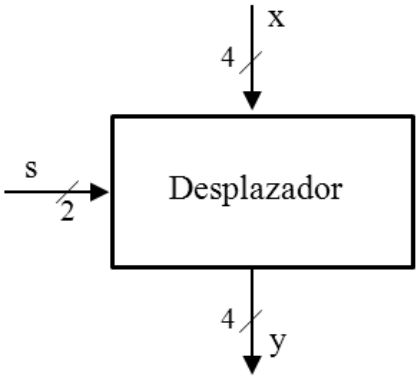
Código VHDL 1.5: Solución al Apartado 1.d: **architecture** del circuito describiendo su estructura

```
--------------------------------------
-- Banco de pruebas
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_alarma is
end entity bp_alarma;
architecture bp_alarma of bp_alarma is
    signal y : std_logic; -- Conectar salidas UUT
    signal x0, x1, x2 : std_logic; -- Conectar entradas UUT
    component alarma is port
          ( y : out std_logic;
            x0, x1, x2 : in std_logic);
    end component alarma;
begin
    -- Instanciar y conectar UUT
    uut : component alarma port map
          ( y => y, x0 => x0, x1 => x1, x2 => x2 );
    gen_vec_test : process
       variable test_in : unsigned (2 downto 0); -- Vector de test
    begin
       test_in := "B000";
       for count in 0 to 7 loop
            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_alarma;
--------------------------------------
```

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

EJERCICIO 2

En la Figura 1.3 se muestra el símbolo lógico de un circuito combinacional desplazador junto con su tabla de operaciones. Obsérvese que este circuito tiene una entrada de selección `s` de dos bits, que determina la operación que realiza el circuito sobre la señal de entrada `x` para obtener la señal de salida `y`.



s(1)	s(0)	Operación
0	0	No modifica el número
0	1	Desplaza a la izquierda rellenando con 0
1	0	Desplaza a la derecha rellenando con 0
1	1	Rota a la derecha

Figura 1.3: Entradas y salidas, y tabla de operaciones del circuito desplazador.

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

```
entity shifter is
   port ( y  : out std_logic_vector(3 downto 0);
          x  : in  std_logic_vector(3 downto 0);
          s  : in  std_logic_vector(1 downto 0) );
end entity shifter;
```

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

2.b) (1 punto) Diseñe en VHDL la **architecture** que describe el comportamiento de un multiplexor 4 a 1.

La **entity** del multiplexor ha de ser la siguiente:

```
entity mux is
   port ( y  : out std_logic;
          x  : in  std_logic_vector(3 downto 0);
          s  : in  std_logic_vector(1 downto 0) );
end entity mux;
```

2.c) (1 punto) Diseñe en VHDL la **architecture** que describe la estructura del circuito combinacional desplazador, la cual debe estar compuesta mediante la conexión de multiplexores 4 a 1 como el diseñado al resolver el Ejercicio 2.b.

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

Solución al Ejercicio 2

La **architecture** describiendo el comportamiento del circuito desplazador se muestra en Código VHDL 1.7.

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

architecture comportamiento of shifter is
begin
   process(s, x)
   begin
      case s  is
      when "00"  => -- pasa sin modificar
         y  <= x;
      when "01"  => -- desplaza izqda con 0
         y  <= x(2  downto 0) & '0';
      when "10"  => -- desplaza drcha con 0
         y  <= '0'  & x(3  downto 1);
      when others => -- rota drcha
         y  <= x(0) & x(3  DOWNTO 1);
      end case;
   end process;
end architecture comportamiento;
-------------------------------------
```

Código VHDL 1.7: Solución al Apartado 2.a: **architecture** del circuito desplazador describiendo su comportamiento.

Un posible diseño del multiplexor 4 a 1 se muestra en Código VHDL 1.8.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
entity mux  is
   port ( y    : out std_logic;
          x    : in    std_logic_vector(3  downto 0);
          s    : in    std_logic_vector(1  downto 0) );
end entity mux;
architecture mux  of mux  is
begin
    y     <= x(0) when (s(1)='0'  and s(0)='0') else
             x(1) when (s(1)='0'  and s(0)='1') else
             x(2) when (s(1)='1'  and s(0)='0') else
             x(3);
end architecture mux;
-------------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.b: multiplexor 4 a 1.

La **architecture** describiendo la estructura del circuito desplazador se muestra en Código VHDL 1.9.

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

architecture estructura of shifter is
component mux is
   port ( y   : out std_logic;
          x   : in   std_logic_vector(3 downto 0);
          s   : in   std_logic_vector(1 downto 0) );
end component mux;
signal m3, m2, m1, m0: std_logic_vector(3 downto 0);
begin
   m3 <= x(0)&'0'&x(2)& x(3);
   m2 <= x(3)&x(3)&x(1)&x(2);
   m1 <= x(2)&x(2)&x(0)&x(1);
   m0 <= x(1)&x(1)&'0'&x(0);
   mux3: mux port map (y(3), m3, s );
   mux2: mux port map (y(2), m2, s);
   mux1: mux port map (y(1), m1, s);
   mux0: mux port map (y(0), m0, s);
end architecture estructura;
-------------------------------------
```

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

Finalmente, el banco de pruebas del circuito desplazador se muestra en Código VHDL 1.10–1.11.

```
--------------------------------------------------
-- Banco de pruebas del 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 salida   : std_logic_vector(3 downto 0); --Salida UUT
    signal op       : std_logic_vector(1 downto 0); --Entradas UUT
    signal entrada  : std_logic_vector(3 downto 0);

    component shifter is
    port ( y    : out std_logic_vector(3 downto 0);
           x    : in  std_logic_vector(3 downto 0);
           s    : in  std_logic_vector(1 downto 0) );
    end component shifter;
  -- Procedure que calcula la salida (expected_s) y lo compara con el
  -- valor de salida que se pasa como argumento (actual_s)
  -- Si ambos valores no coinciden, se muestra un mensaje y se
  -- incrementa el contador de errores (error_count)
  procedure check_desp
      ( op          : in integer;
        data        : in unsigned(3 downto 0);
        actual_s    : in unsigned (3      downto 0);
        error_count: inout integer ) is
    variable expected_s : unsigned(3 downto 0);
  begin
    case op is
         when 0       => expected_s := data;
         when 1       => expected_s := data sll 1;
         when 2       => expected_s := data srl 1;
         when others => expected_s := data ror 1;
    end case;
        assert( expected_s = actual_s)
        report "ERROR. Entrada: " & integer'image(to_integer(data)) &
                ", Operacion: " & integer'image(op) &
                ", resultado esperado: " &
                integer'image(to_integer(expected_s)) &
                ", resultado actual: " &
                integer'image(to_integer(actual_s)) &
                " en el instante "&
                time'image(now);

    if (expected_s /= actual_s) then
      error_count := error_count + 1;
    end if;

  end procedure check_desp;
  -- Fin de la definición del procedure
```

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

```
begin
   UUT : component shifter port map
          (salida, entrada, op);

vec_test : process is
      variable temp   : std_logic_vector (5 downto 0);
      variable error_count : integer := 0;
   begin
      report "Comienza la simulación";
      -- Generar todos los posibles valores de entrada
      for i in 0 to 2**6-1 loop
         temp := std_logic_vector(to_unsigned(i,6));
         op    <= temp(5 downto 4);
         entrada    <= temp(3 downto 0);
         wait for DELAY;

         check_desp(to_integer(unsigned(op)),          unsigned(entrada),
unsigned(salida), error_count);

      end loop;

        -- Informe mostrando el resultado del test
      report "Finaliza la simulación: " & integer'image(error_count) & "
errores";

      wait;    -- Termina 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.