

INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2013

EJERCICIO 1

Dada las dos funciones lógicas (F y G) mostradas a continuación, que dependen de 3 variables (x, y y z):

$$F = \text{not } x \text{ or } y \text{ and } z$$

$$G = x \text{ and } y \text{ and } z$$

- 1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito.
- 1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.
- 1.c) (0.5 puntos) Dibuje el diagrama de un circuito que implemente estas dos funciones lógicas 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.
- 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, las salidas 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 se muestra en Código VHDL 1.1.

```
-----
library IEEE;
use IEEE.std_logic_1164.all;
entity funcLog_F_G is
    port ( F, G      : out std_logic;
           x, y, z   : in  std_logic );
end entity funcLog_F_G;
-----
```

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

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

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

architecture funcLog_F_G_Comp of funcLog_F_G is
begin
    F <= not x or (y and z);
    G <= x and y and z;
end architecture funcLog_F_G_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 empleando dos puertas AND, un inversor y una puerta OR.

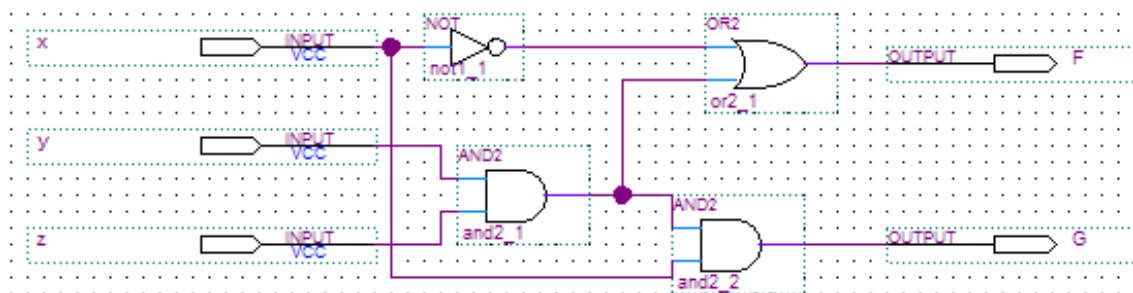


Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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, respectivamene.

```

-----
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.

```

-----
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 1.5: Puerta OR lógica.

El Código VHDL 1.6 muestra la **architecture** del circuito describiendo estructura.

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.7.

```

-----
library IEEE;
use IEEE.std_logic_1164.all;
architecture funcLog_F_G_Estruc of funcLog_F_G is
    signal not_x, yz: 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
    not1_1 : component not1 port map (not_x, x);
    and2_1 : component and2 port map (yz, y, z);
    or2_1 : component or2 port map (F, not_x, yz);
    and2_2 : component and2 port map (G, x, yz);
end architecture funcLog_F_G_Estruc;
-----

```

Código VHDL 1.6: 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_funcLog_F_G is
end entity bp_funcLog_F_G;

architecture bp_funcLog_F_G of bp_funcLog_F_G is
    signal F, G : std_logic; -- Conectar salidas UUT
    signal x0, x1, x2 : std_logic; -- Conectar entradas UUT

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

begin
    -- Instanciar y conectar UUT
    uut : component funcLog_F_G port map
        ( F => F, G => G,
          x => x0, y => x1, z => x2);

    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
            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_funcLog_F_G;
-----

```

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

EJERCICIO 2

En la Figura 1.2 se muestra el símbolo lógico de un circuito combinacional que tiene dos entradas a y b de 8 bits y una salida y de cuatro bits. El valor de la señal de salida y indica el número de posiciones en que difieren las dos señales de entrada (a y b). Por ejemplo, para las señales de entrada “00010011” y “10010010”, que difieren únicamente en los bits de las posiciones 0 y 7, el circuito genera la señal de salida “0010” (valor 2 en decimal).

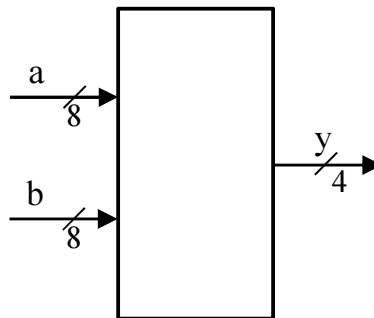


Figura 1.2: Entradas y salidas del circuito combinacional.

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

```
entity circ is
  port ( y : out std_logic_vector(3 downto 0);
        a : in  std_logic_vector(7 downto 0);
        b : in  std_logic_vector(7 downto 0) );
end entity circ;
```

- 2.a) (3 puntos) Diseñe en VHDL la **architecture** que describe el comportamiento del circuito combinacional descrito en el enunciado.
- 2.b) (3 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito. 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 al Apartado 2.a.

Solución al Ejercicio 2

La **architecture** describiendo el comportamiento del circuito combinacional se muestra en Código VHDL 1.8.

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

architecture circ of circ is
signal aux: unsigned (7 downto 0);
signal lev0_0, lev0_2, lev0_4 , lev0_6: unsigned (1 downto 0);
signal lev1_0, lev1_4 : unsigned(2 downto 0);
signal lev2: unsigned (3 downto 0);
begin
    aux <= unsigned ( a xor b);
    lev0_0 <= ( '0' & aux(0)) + ( '0' & aux(1) );
    lev0_2 <= ( '0' & aux(2)) + ( '0' & aux(3) );
    lev0_4 <= ( '0' & aux(4)) + ( '0' & aux(5) );
    lev0_6 <= ( '0' & aux(6)) + ( '0' & aux(7) );
    lev1_0 <= ( '0' & lev0_0) + ( '0' & lev0_2 );
    lev1_4 <= ( '0' & lev0_4) + ( '0' & lev0_6 );
    lev2 <= ( '0' & lev1_0) + ( '0' & lev1_4);
    y <= std_logic_vector( lev2 );
end architecture circ;
-----
```

Código VHDL 1.8: Solución al Apartado 2.a: **architecture** del circuito combinacional describiendo su comportamiento.

El banco de pruebas del circuito se muestra en Código VHDL 1.9–1.10.

```

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

entity bp_circ is
end entity bp_circ;

architecture bp_circ of bp_circ is
    signal y      : std_logic_vector(3 downto 0); -- Conectar salidas UUT
    signal a, b    : std_logic_vector(7 downto 0); -- Conectar entradas UUT

    component circ is
        port ( y : out std_logic_vector(3 downto 0);
              a : in  std_logic_vector(7 downto 0);
              b : in  std_logic_vector(7 downto 0) );
    end component circ;

begin
    -- Instanciar y conectar UUT
    uut : component circ
        port map( y, a, b );

```

Código VHDL 1.9: Solución al Apartado 2.b: banco de pruebas del circuito.


```

gen_vec_test : process
  variable test_ina      : unsigned (7 downto 0); -- Vector de test
  variable test_inb      : unsigned (7 downto 0); -- Vector de test
  variable dif           : std_logic_vector(7 downto 0);
  variable sum           : unsigned (3 downto 0);
  variable esperado_Y    : std_logic_vector(3 downto 0);
  variable error_count   : integer := 0;
begin
  test_ina := B"00000000";
  report "Comienza la simulación";
  for counta in 0 to 255 loop
    test_inb := B"00000000";
    for countb in 0 to 255 loop
      a <= std_logic_vector(test_ina);
      b <= std_logic_vector(test_inb);
      dif := std_logic_vector(test_ina) xor std_logic_vector(test_inb);
      sum := B"0000";
      for i in 0 to 7 loop
        sum := sum + ('0' & dif(i));
      end loop;
      esperado_Y := std_logic_vector(sum);
      wait for 10 ns;
      test_inb := test_inb + 1;
      if (esperado_Y /= y ) then
        report "ERROR en la salida valida. Valor esperado: " &
          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(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;
    test_ina := test_ina + 1;
  end loop;
  report "ERROR: Hay " &
    integer'image(error_count) &
    " errores.";

  wait;
end process gen_vec_test;
end architecture bp_circ;
-----

```

Código VHDL 1.10: Solución al Apartado 2.b: continuación del banco de pruebas del circuito.