# INGENIERÍA DE COMPUTADORES 3

## Solución al Trabajo Práctico - Junio de 2015

### EJERCICIO 1

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

| x | y | z | F | G |
|---|---|---|---|---|
| '0' | '0' | '0' | '0' | '1' |
| '0' | '0' | '1' | '1' | '1' |
| '0' | '1' | '0' | '0' | '0' |
| '0' | '1' | '1' | '1' | '1' |
| '1' | '0' | '0' | '0' | '0' |
| '1' | '0' | '1' | '0' | '0' |
| '1' | '1' | '0' | '1' | '0' |
| '1' | '1' | '1' | '1' | '0' |

**1.a)** (0.5 puntos) Obtenga las funciones lógicas `F` y `G` a partir de la tabla de verdad. Escriba en VHDL la **entity** del circuito que implemente las dos funciones lógicas. Es decir, que tenga tres entradas `x`, `y` y `z`, y dos salidas `F` y `G`.

**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. 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)** (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. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas.

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

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

```
-------------------------------------
architecture Comp of funcFG is
begin
 F <= (x and y) or (not x and z);
 G <= (not x and not y) or (not 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 tres puertas AND de dos entradas, dos puertas OR de dos entradas y dos inversores.

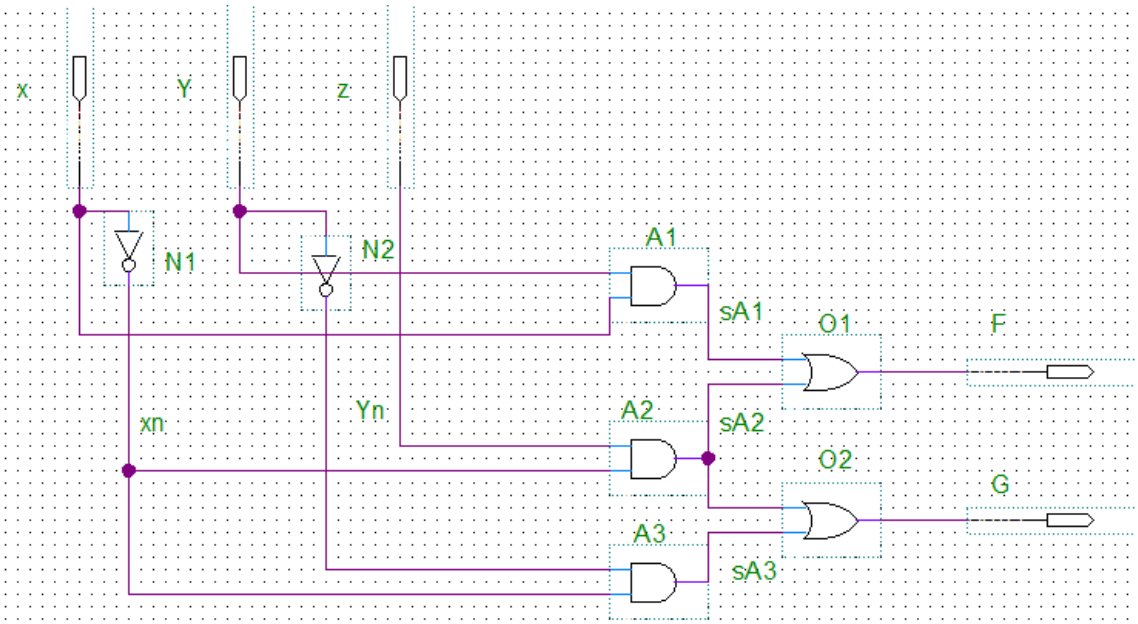


**Figura 1.1:** Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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, respectivamente. El Código VHDL 1.6 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 1.3:** 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;

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 1.5:** Puerta NOT lógica.

```
-----------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcFG is
   signal xn, yn: std_logic;
   signal sA1, sA2, sA3: 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);
   A1 : component and2 port map (sA1, x, y);
   A2 : component and2 port map (sA2, xn, z);
   A3 : component and2 port map (sA3, xn, yn);
   O1 : component or2 port map (F, sA1, sA2);
   O2 : component or2 port map (G, sA3, sA2);
end architecture circuito_Estruc;
---------------------------------
```

**Código VHDL 1.6:** 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 1.7. 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_funcFG is
    constant DELAY      : time      := 20 ns; -- Retardo usado en el test
end entity bp_funcFG;

architecture bp_funcFG of bp_funcFG is
    signal F, G : std_logic;
    signal x, y, z    : std_logic;

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

begin
    UUT : component funcFG port map
          (F, G, 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_funcFG;
--------------------------------------------------
```

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

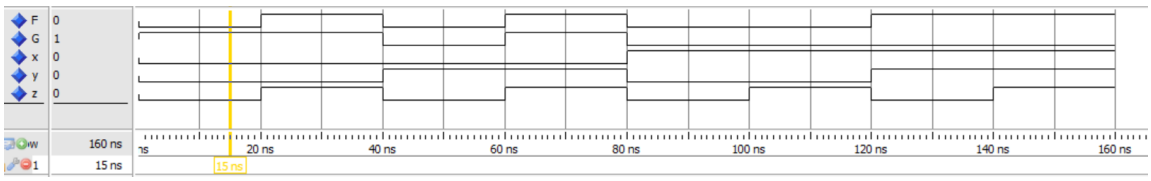


**Figura 1.2:** Cronograma del Apartado 1.e.

## EJERCICIO 2

Se desea diseñar un circuito digital combinacional con una señal de entrada de 8 bits llamada `data`, una señal de salida de 4 bits llamada `total` y una señal de salida de 1 bit llamada `error`.

El circuito ha de tener el siguiente comportamiento:

- Si no existe una secuencia de ceros en la señal de entrada (es decir, si `data` tiene el valor "11111111"), la señal `error` ha de valer '0' y la señal `total` ha de valer "0000".
- Si existe una única secuencia de ceros en la señal de entrada, la señal de salida `error` ha de valer '0' y la señal `total` ha de tener como valor el número de ceros en la secuencia de ceros. Una secuencia de ceros es uno o más bits '0' en posiciones consecutivas. Por ejemplo:
  - El valor de la señal de entrada `data` "11000001" da como resultado un valor de la señal `error` de '0' y un valor de la señal `total` de "0101".
  - El valor de la señal de entrada `data` "11011111" da como resultado un valor de la señal `error` de '0' y un valor de la señal `total` de "0001".
- Si existen dos o más secuencias de ceros, el valor de la señal `error` es '1' y el valor de la señal `total` es "0000". Por ejemplo, el valor de la señal de entrada `data` "00111110" da como resultado un valor de la señal `error` de '1' y un valor de la señal `total` de "0000".

**2.a)** (3 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito digital combinacional descrito. Puede emplear en el diseño del circuito la sentencia **exit**.

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

```
entity contador is
  port ( error: out std_logic;
         total: out std_logic_vector( 3 downto 0 );
         data:  in  std_logic_vector( 7 downto 0 ) );
end entity contador;
```

**2.b)** (3 puntos) Programe en VHDL un banco de pruebas que testee el circuito digital. El banco de pruebas debe generar los 10 vectores de test siguientes:

- Valor de la señal de entrada `data` “00000000”.
- Valor de la señal de entrada `data` “11111111”.
- Valor de la señal de entrada `data` “00011000”.
- Valor de la señal de entrada `data` “11100000”.
- Valor de la señal de entrada `data` “00000111”.
- Valor de la señal de entrada `data` “01111100”.
- Valor de la señal de entrada `data` “01111110”.
- Valor de la señal de entrada `data` “01001101”.
- Valor de la señal de entrada `data` “10000001”.
- Valor de la señal de entrada `data` “11110001”.

Defina en el banco de pruebas los tres tipos vectoriales siguientes:

```
type vector_test is array (0 to 9) of std_logic_vector(7 downto 0);
type error_array is array (0 to 9) of std_logic;
type total_array is array (0 to 9) of std_logic_vector(3 downto 0);
```

Los valores de los vectores de test y los valores esperados de las señales `error` y `total` se han de almacenar en constantes de estos tipos vectoriales. La asignación de valor a la señal de entrada del circuito bajo test se ha de realizar dentro de un bucle **for**.

El banco de pruebas ha de mostrar un mensaje de error en caso de que las salidas obtenidas de la UUT no correspondan con las esperadas, indicando la posición del vector de test (0 a 9) que no ha producido la salida correcta. 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.

## Solución al Ejercicio 2

La **architecture** describiendo el comportamiento del circuito se muestra en Código VHDL 1.8. El banco de pruebas se muestra en Código VHDL 1.9. El cronograma obtenido al simular el banco de pruebas se muestra en la Figura 1.3.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture contador of contador is
begin
  process(data)
     variable temp_total: unsigned (3 downto 0);
     --La var. inicio_sec se pone a 1 al detectar
     --el inicio de la primera secuencia de ceros
     variable inicio_sec: std_logic;
     --La variable fin_sec se pone a 1 al detectar
     --el final de la primera secuencia de ceros
     variable fin_sec: std_logic;
  begin
     error <= '0';
     inicio_sec := '0';
     fin_sec := '0';
     temp_total := (others => '0');
     for i in 0 to 7 loop
        if (fin_sec = '1' and data (i) = '0') then
           temp_total := (others => '0');
           error <= '1';
           exit;
        elsif (inicio_sec = '1' and data(i) = '1') then
           fin_sec := '1';
        elsif (data(i) = '0') then
           inicio_sec := '1';
           temp_total := temp_total + 1;
        end if;
     end loop;
     total <= std_logic_vector(temp_total);
  end process;
end contador;
-------------------------------------
```

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

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

entity bp_contador is
end entity bp_contador;

architecture bp_contador of bp_contador is
   signal total : std_logic_vector (3 downto 0); -- Conectar salidas UUT
   signal error : std_logic;
   signal data: std_logic_vector(7 downto 0); -- Conectar entradas UUT

   component contador is port
    ( error: out std_logic;
      total: out std_logic_vector (3 downto 0);
      data : in std_logic_vector(7 downto 0));
   end component contador;
   type vector_test is array (0 to 9) of std_logic_vector( 7 downto 0);
   type error_array is array (0 to 9) of std_logic;
   type total_array is array (0 to 9) of std_logic_vector(3 downto 0);
   constant data_const: vector_test :=
    ( "00000000", "11111111", "00011000", "11100000", "00000111",
      "01111100", "01111110","01001101","10000001","11110001");
   constant error_const:error_array :=
     ('0', '0', '1', '0', '0', '1', '1', '1', '0', '0');
   constant total_const: total_array :=
    ("1000", "0000", "0000", "0101", "0101", "0000", "0000",
     "0000", "0110", "0011");

begin
-- Instanciar y conectar UUT
  uut : component contador port map
  ( error, total, data);

  gen_vec_test : process
   variable n_err : integer := 0; -- Vector de test
  begin
   for i in 0 to 9 loop
     data <= data_const(i);
     wait for 10 ns;
     if(error /= error_const(i)) or(total /= total_const(i)) then
        report("Error en el dato ("&
               integer'image(i) & ").");
             n_err := n_err + 1;
     end if;
   end loop;
   report ( "Existen " & integer'image(n_err) & " errores.");
   wait;
  end process gen_vec_test;
end architecture bp_contador;
-------------------------------------
```

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

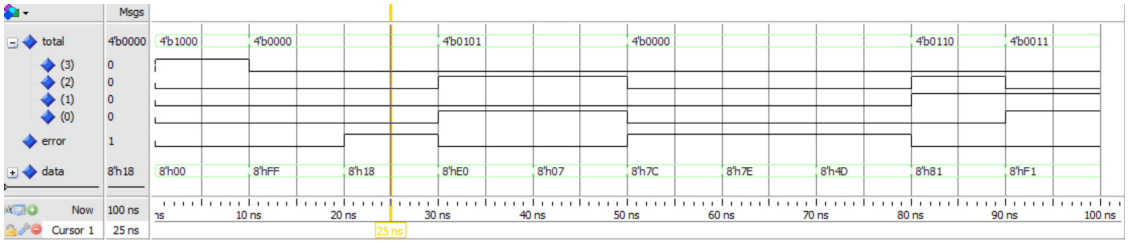


**Figura 1.3:** Cronograma del Apartado 2.b.