INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2022

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.

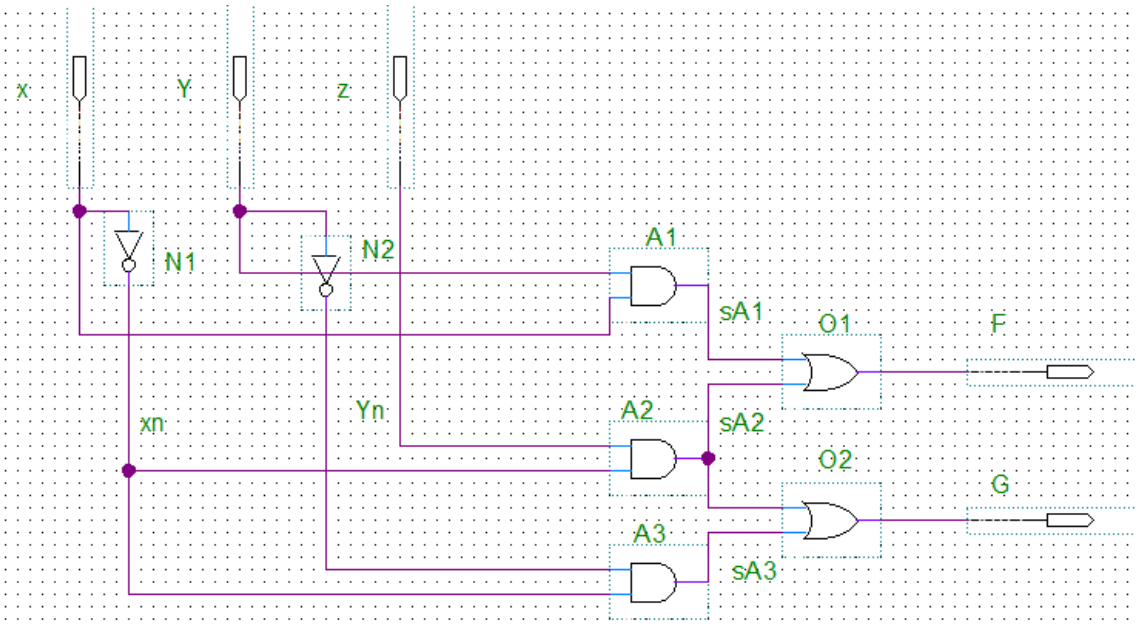

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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.

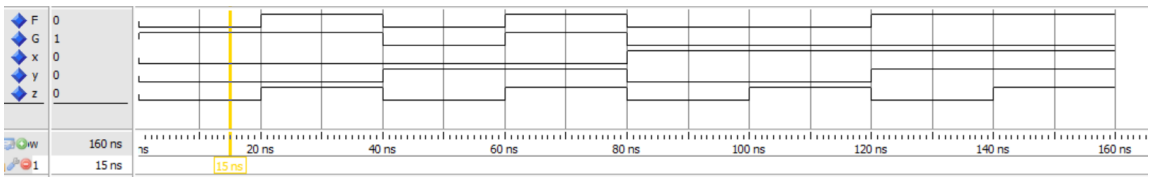

Figura 1.2: Cronograma del Apartado 1.e.

EJERCICIO 2

Se quiere programar en VHDL un circuito combinacional que realiza la suma con acarreo de entrada de dos número binarios con signo (representados en complemento a 2) y además proporciona tres señales de salida que informan de si ha habido desbordamiento en la suma, si el resultado es cero y el signo que debiera tener el resultado. El circuito tiene las siguientes señales de entrada: los operandos de n bits `a` y `b` y una señal de acarreo de entrada un bit llamada `cin`. El circuito tiene las siguientes señales de salida: el resultado de n bits `res` y las señales de 1 bit `desbordamiento`, `cero` y `signo`.

El valor de la señal `desbordamiento` es '1' solo si la operación de suma produce desbordamiento. En cualquier otro caso su valor es '0'. Para el cálculo de esta señal ha de tener en cuenta lo siguiente:

- Si los dos operandos tienen diferentes signos, no puede existir desboradamiento ya que la suma de un número positivo y un número negativo siempre provoca una disminución de la magnitud.
- Si los dos operandos y el resultado tienen el mismo signo, no ocurre desbordamiento ya que el resultado está dentro del mismo rango.
- Si los dos operandos tienen el mismo signo pero el resultado tiene un signo diferente, está ocurriendo desbordamiento. El cambio de signo indica que el resultado va más allá del límite positivo o negativo y está, por tanto, más allá del rango.

Para el cálculo de la señal de desbordamiento ha de traducir estas tres observaciones en una expresión lógica.

El valor de la señal `cero` es '1' solo si el resultado de la operación tiene valor cero y, además, no existe desbordamiento. En cualquier otro caso su valor es '0'.

El valor de la señal `signo` es el mismo bit de signo del resultado de la suma solo si no existe desbordamiento. En caso de existir desbordamiento, el signo es el inverso al resultado de la suma.

2.a) (3 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento del circuito combinacional empleando solo sentencias concurrentes. Los nombres de los puertos de la **entity** deber ser los mismos que se han especificado para las señales de entrada y salida del circuito. Emplee

el convenio de especificar en primer lugar las señales de salida del circuito y posteriormente las señales de entrada. El número de bits (n) de las señales `a`, `b` y `res` se ha de expresar como constante del tipo **generic**.

En el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

```
IEEE.std_logic_1164
IEEE.numeric_std
```

2.b) (3 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito diseñado en el Apartado 2.a. El número de bits de los operandos de entrada ha de ser una constante del programa cuyo valor se pueda modificar de modo que el banco de pruebas sea válido para cualquier número de bits de los operandos de entrada.

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. El banco de pruebas debe mostrar al final del test un mensaje con el número total de errores detectados. La forma de calcular el valor esperado de la señal de desbordamiento debe ser diferente a la realizada en el circuito que se pretende testear.

Realice la simulación para el caso en que el número de bits de los operandos sea cuatro (n=4). Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas del circuito diseñado en el Apartado 2.a.

Solución al Ejercicio 2

La **entity** del circuito combinacional se muestra en Código VHDL 1.8.

El Código VHDL 1.9 muestra el código VHDL del circuito combinacional empleando solo sentencias concurrentes.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity sumador_status is
generic (n:integer:=8);
    port ( res       : out    std_logic_vector (n-1  downto 0);
           desbordamiento : out std_logic;
           cero     : out std_logic;
           signo    : out std_logic;
           cin      : in std_logic;
           a,    b     :     in     std_logic_vector     (n-1  downto     0)     );

end entity sumador_status;
```

Código VHDL 1.8: Solución al Apartado 2.a: **entity** del circuito.

```
-------------------------------------------------
-- Sumador de n bits con señales adicionales
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

architecture sumador_status of sumador_status is
    signal a_ext, b_ext, res_ext : signed(n  downto 0);
    signal desb  : std_logic;
    signal sign_a, sign_b, sign_res: std_logic;
begin
    sign_a <= a_ext(n);
    sign_b <= b_ext(n);
    sign_res <= res_ext(n);
    a_ext <= signed( a & '1');
    b_ext <= signed( b & cin);

    res_ext <= a_ext + b_ext;
    desb <= (sign_a  and sign_b  and (not sign_res)) or
                          ( (not sign_a) and (not sign_b) and sign_res);
    signo <= res_ext(n) when (desb = '0') else
            ( not res_ext(n) ) when (desb= '1');
    cero <= '1'  when ((res_ext(n  downto 1) = 0) and (desb = '0')) else
              '0';
    desbordamiento <= desb;
    res <= std_logic_vector(res_ext(n  downto 1));
end architecture sumador_status;
-------------------------------------------------
```

Código VHDL 1.9: Solución al Apartado 2.a: circuito combinacional.

El banco de pruebas del circuito combinacional se muestra en Código VHDL 1.10–1.13. El cronograma obtenido al simular el banco de pruebas usando como circuito a testear el diseño del Apartado 2.a se muestra en la Figura 1.3.

```
-- Banco de pruebas del sumador con señales de status
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_sumador is          -- El banco de pruebas no tiene
end entity bp_sumador;        -- ni entradas ni salidas

architecture bp_sumador of bp_sumador is
   constant WIDTH      :   integer := 4;
   signal res, a, b: std_logic_vector (WIDTH-1 downto 0);
   signal acarreo_in: std_logic;    -- Para conectar el UUT
   signal desbordamiento, cero, signo : std_logic;

   component sumador_status is
   generic (n:integer:=WIDTH);
      port (res       : out    std_logic_vector (n-1 downto 0);
            desbordamiento : out std_logic;
            cero      : out std_logic;
            signo     : out std_logic;
            cin       : in std_logic;
            a, b      : in std_logic_vector (n-1 downto 0));
   end component sumador_status;

begin
   -- Instanciar y conectar UUT
   uut : component sumador_status port map
         (res, desbordamiento, cero, signo, acarreo_in, a, b);
   -- Crear vectores de test y comprobar salidas del UUT
   gen_vec_test : process
    variable sumaEsperada : signed (WIDTH-1 downto 0); -- Resultado suma
    variable suma: integer;
    variable desbEsperado : std_logic;
    variable ceroEsperado : std_logic;
    variable signoEsperado : std_logic;
    variable num_errores   : integer := 0;              -- Numero de errores
    variable aca_in : std_logic_vector(0 downto 0);
   begin
```

Código VHDL 1.10: Solución al Apartado 2.b: banco de pruebas del circuito (1/4).

```
for sumando_X in -2**(WIDTH-1) to 2**(WIDTH-1)-1 loop
  for sumando_Y in -2**(WIDTH-1) to 2**(WIDTH-1)-1 loop
    for ac_in in 0 to 1 loop
      suma := sumando_X+sumando_Y+ac_in;
      sumaEsperada := to_signed(ac_in,WIDTH) +
                      to_signed(sumando_x,WIDTH)  +
                      to_signed(sumando_y,WIDTH);
      a  <= std_logic_vector(to_signed(sumando_x,WIDTH));
      b    <= std_logic_vector(to_signed(sumando_y,WIDTH));
      aca_in    := std_logic_vector(to_unsigned(ac_in,1));
      acarreo_in <= aca_in(0);
      if (suma>(2**(WIDTH-1)-1) or (suma<-2**(WIDTH-1)) ) then
          desbEsperado := '1';
      else
          desbEsperado := '0';
      end if;
      if (suma=0) and (desbEsperado='0') then
           ceroEsperado := '1';
      else
           ceroEsperado := '0';
      end if;
      if ((suma<0) ) then
        signoEsperado := '1';
      else
           signoEsperado := '0';
      end if;
        wait for 100 ns;

      if (std_logic_vector(sumaEsperada) /= res) then
          num_errores := num_errores + 1;
          report time'image(now)   &
              ": "                 &
            -- std_logic'image(sumaEsperada(4)) &
              std_logic'image(sumaEsperada(3)) &
              std_logic'image(sumaEsperada(2)) &
              std_logic'image(sumaEsperada(1)) &
              std_logic'image(sumaEsperada(0)) &
              " Suma calculada :"              &
            -- std_logic'image(acarreo_out)          &
              std_logic'image(res(3))             &
              std_logic'image(res(2))             &
              std_logic'image(res(1))             &
              std_logic'image(res(0))             ;

              end if;
```

Código VHDL 1.11: Solución al Apartado 2.b: banco de pruebas del circuito (2/4).

```
        if (desbEsperado /= desbordamiento) then
              num_errores := num_errores + 1;
              report "Error en el desbordamiento";
              report time'image(now)   &
                  " Suma calculada :"              &
                  std_logic'image(res(3))          &
                  std_logic'image(res(2))          &
                  std_logic'image(res(1))          &
                  std_logic'image(res(0))      &
                  " desbordamiento :"              &
                  std_logic'image(desbordamiento) &
                 " desbordamiento esperado:"              &
                  std_logic'image(desbEsperado) &

                  " a :"              &
                  std_logic'image(a(3)) &
                  std_logic'image(a(2)) &
                  std_logic'image(a(1)) &
                  std_logic'image(a(0)) &
                  " b :"              &
                  std_logic'image(b(3))            &
                  std_logic'image(b(2))            &
                  std_logic'image(b(1))            &
                  std_logic'image(b(0))            ;

          end if;
          if (ceroEsperado /= cero) then
              num_errores:= num_errores + 1;
              report "Error en el cero";
              report time'image(now)   &
                  " cero :"              &
                  std_logic'image(cero) &

                  " a :"              &
                  std_logic'image(a(3)) &
                  std_logic'image(a(2)) &
                  std_logic'image(a(1)) &
                  std_logic'image(a(0)) &
                  " b :"              &
                  std_logic'image(b(3))            &
                  std_logic'image(b(2))            &
                  std_logic'image(b(1))            &
                  std_logic'image(b(0))            ;
        end if;
```

Código VHDL 1.12: Solución al Apartado 2.b: continuación del banco de pruebas del circuito (3/4).

```
            if (signoEsperado /= signo) then
                num_errores := num_errores + 1;
                report "Error en el signo";
                report time'image(now)   &
                   " signo :"              &
                   std_logic'image(signo) &
                  " desbordamiento :"              &
                   std_logic'image(desbordamiento) &

                   " a :"              &
                   std_logic'image(a(3)) &
                   std_logic'image(a(2)) &
                   std_logic'image(a(1)) &
                   std_logic'image(a(0)) &
                   " b :"              &
                   std_logic'image(b(3))              &
                   std_logic'image(b(2))              &
                   std_logic'image(b(1))              &
                   std_logic'image(b(0))              ;
            end if;
          end loop;
        end loop;
    end loop;
    report "Test completo. Hay "      &
            integer'image(num_errores) &
            " errores.";
    wait;
  end process gen_vec_test;
end architecture bp_sumador;
-------------------------------------------------
```

Código VHDL 1.13: Solución al Apartado 2.b: continuación del banco de pruebas del circuito (4/4).

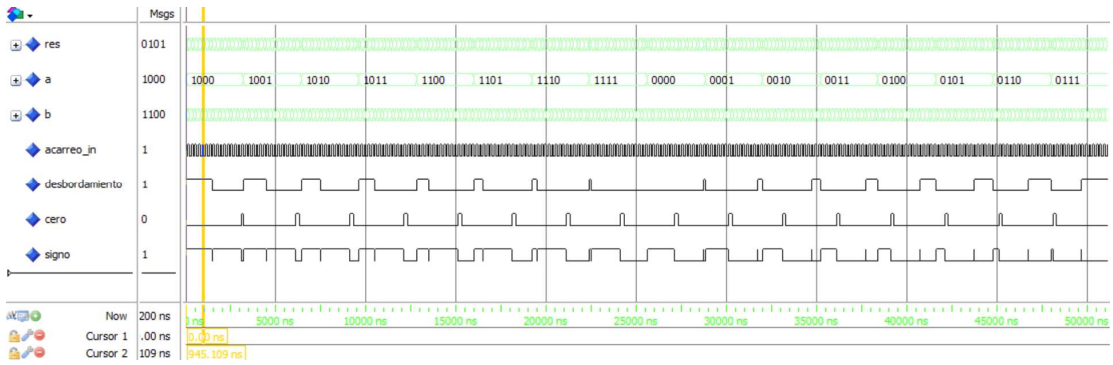

Figura 1.3: Cronograma del Apartado 2.b comprobando el comportamiento del circuito diseñado en el Apartado 2.a.