INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Convocatoria ordinaria 2023

Ejercicio 1

A continuación se muestran dos funciones lógicas `F` y `G`, que dependen de las cuatro variables `A`, `B`, `C` y `D` de la forma mostrada a continuación:

```
F = A B C + C D
G = A B C'
```

1.a) (0.25 puntos) Escriba en VHDL la **entity** de un circuito que implemente las dos funciones lógicas. Es decir, que tenga cuatro entradas `A`, `B`, `C` y `D`, y dos salidas `F` y `G`. Emplee el convenio de especificar en primer lugar las señales de salida del circuito y posteriormente las señales de entrada.

1.b) (0.75 puntos) Escriba en VHDL la **architecture** que describa el *comportamiento* de un circuito que implemente las dos funciones lógicas.

1.c) (0.5 puntos) Dibuje el diagrama al nivel de puertas lógicas de un circuito que implemente estas dos funciones lógicas (no es necesario simplificar dichas funciones). Emplee para ello puertas lógicas AND y OR de dos entradas, y una 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, 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 los dos cronogramas obtenidos al realizar la simulación del banco de pruebas con los circuitos diseñados en los Apartados 1.b y 1.d.

Solución al Ejercicio 1

El Código VHDL 1.1 y 1.2 es una posible solución a los Apartados 1.a y 1.b.

El diagrama al nivel de puertas lógicas pedido en el Apartado 1.c está representado en la Figura 1.1. Obsérvese que en la figura se han señalado las señales `ab`, `abc`, `cd` y `not_c`, que se usarán en la descripción estructural. El Código VHDL 1.3 – 1.5 es una posible descripción de las puertas lógicas AND y OR de dos entradas, y la puerta NOT. El Código VHDL 1.6 es una posible descripción estructural del circuito, que corresponde con el diagrama mostrado en la Figura 1.1.

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

entity funcLog_F_G is
   port ( F, G       : out std_logic;
          a, b, c, d : in  std_logic );
end entity funcLog_F_G;
```

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

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

architecture funcLog_F_G_Comp of funcLog_F_G is
begin
   F <= (a and b and c) or (c and d);
   G <= a and b and not c;
end architecture funcLog_F_G_Comp;
```

Código VHDL 1.2: Apartado 1.b: **architecture** que describe el comportamiento.

```vhdl
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: Apartado 1.c: puerta AND de dos entradas.

```vhdl
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: Apartado 1.c: puerta OR de dos entradas.

```vhdl
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: Apartado 1.c: puerta NOT con una entrada.

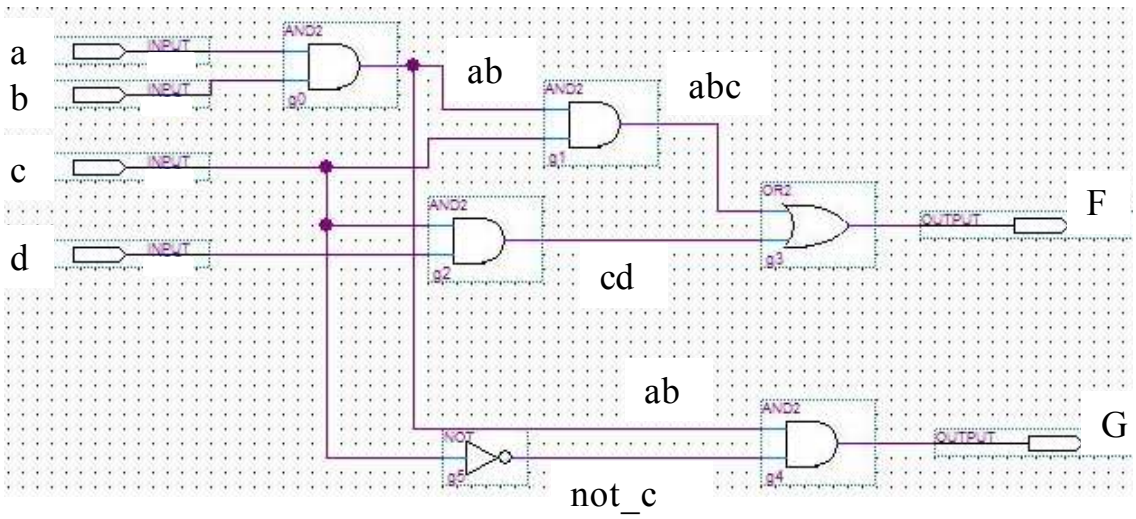

Figura 1.1: Apartado 1.c: diagrama al nivel de puertas lógicas.

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

architecture funcLog_F_G_Estruc of funcLog_F_G is
   signal not_c, ab, abc, cd: 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 and2 port map (ab, a, b);
   g1 : component and2 port map (abc, ab, c);
   g2 : component and2 port map (cd, c, d);
   g3 : component or2  port map (F, abc, cd);
   g4 : component and2 port map (G, ab, not_c);
   g5 : component not1 port map (not_c, c);
end architecture funcLog_F_G_Estruc;
```

Código VHDL 1.6: Apartado 1.d: descripción estructural al nivel de puertas lógicas.

Finalmente, el Código VHDL 1.7 es un banco de pruebas para los diseños del circuito que implementa las dos funciones lógicas. En las Figuras 1.2–1.3 se muestran las formas de onda obtenidas al simular el banco de pruebas con los dos diseños anteriormente realizados. La comprobación de que el diseño funciona correctamente debe hacerse en este caso mediante inspección visual.

```vhdl
-- 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 y0, y1         : std_logic;  -- Conectar salidas UUT
   signal x0, x1, x2, x3 : std_logic;  -- Conectar entradas UUT

   component funcLog_F_G is
      port ( F, G       : out std_logic;
             a, b, c, d : in std_logic);
   end component funcLog_F_G;

begin
   -- Instanciar y conectar UUT
   uut : component funcLog_F_G port map
         ( F => y0, G => y1,
           a => x0, b => x1, c => x2, d => x3 );

   gen_vec_test : process
      variable test_in : unsigned (3 downto 0); -- Vector de test
   begin
      test_in := B"0000";
      for count in 0 to 15 loop
         x3 <= test_in(3);
         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: Apartado 1.e: banco de pruebas.

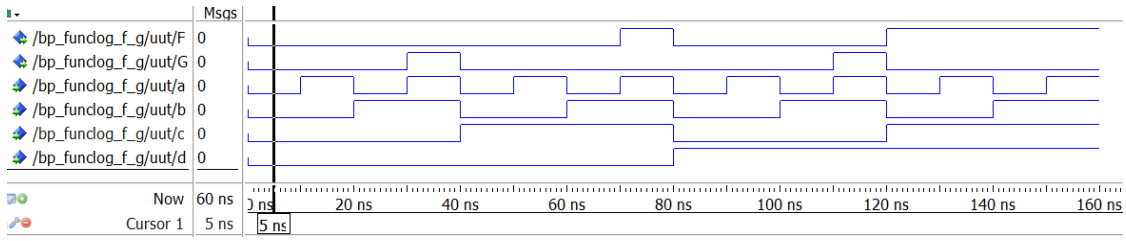

Figura 1.2: Apartado 1.e: simulación del banco de pruebas del diseño describiendo el comportamiento del circuito.

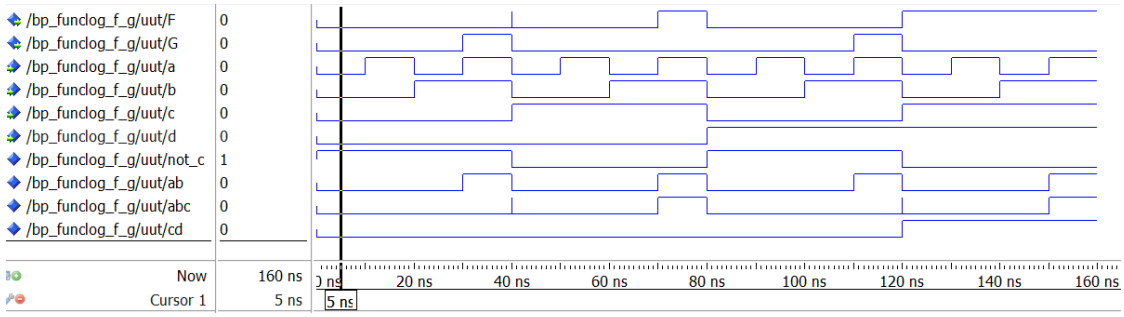

Figura 1.3: Apartado 1.e: simulación del banco de pruebas del diseño describiendo la estructura del circuito.

Ejercicio 2

Se quiere programar en VHDL un circuito combinacional que realiza la suma o la resta (dependiendo del valor de una señal de control) de dos número binarios con signo (representados en complemento a 2). El circuito tiene las siguientes señales de entrada: los operandos de 8 bits `a` y `b`, y una señal de entrada de un bit llamada `ctrl`. El circuito tiene la siguiente señal de salida: el resultado de 8 bits `y`.

El valor de la señal `ctrl` especifica la operación deseada. Si el valor de `ctrl` es '0', el valor de la señal de salida es el resultado de realizar la operación suma sobre los dos operandos de entrada. Por el contrario, si el valor de la señal `ctrl` es '1', el valor de la señal de salida `y` es el resultado de realizar la operación de resta `a-b`.

2.a) (0.25 puntos) Escriba en VHDL la **entity** del circuito combinacional descrito anteriormente. 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.

2.b) (0.75 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito combinacional empleando solo sentencias de asignación concurrentes, que pueden ser simples y/o condicionales. Además, emplee los operadores aritméticos suma y resta definidos para los tipos de dato **signed**.

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

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

2.c) (3 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito combinacional empleando solo sentencias de asignación concurrentes, que pueden ser simples y/o condicionales. En este diseño no se puede emplear el operador aritmético resta, pudiendo únicamente emplear el operador aritmético suma definido para los tipos de dato **signed**. Para realizar este diseño, tenga en cuenta que en la representación en complemente a dos de un número, la resta $a - b$ puede calcularse indirectamente como $a + \bar{b} + 1$, siendo $\bar{b}$ el resultado de invertir bit a bit el operando b. Esto nos va a permitir realizar el diseño sin emplear un bloque restador.

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

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

2.d) (2.5 puntos) Programe en VHDL un banco de pruebas que testee los circuitos diseñados en los Apartados 2.b y 2.c para los siguientes conjuntos de valores de las señales de entrada:

- Conjunto 1: `a = "00000000"`, `b = "00000000"`, `ctrl = '0'`
- Conjunto 2: `a = "00000000"`, `b = "00000000"`, `ctrl = '1'`
- Conjunto 3: `a = "00000011"`, `b = "00000001"`, `ctrl = '0'`
- Conjunto 4: `a = "00000011"`, `b = "00000001"`, `ctrl = '1'`
- Conjunto 5: `a = "10001000"`, `b = "00010010"`, `ctrl = '0'`
- Conjunto 6: `a = "10001000"`, `b = "00010010"`, `ctrl = '1'`
- Conjunto 7: `a = "11111111"`, `b = "00000001"`, `ctrl = '0'`
- Conjunto 8: `a = "11111111"`, `b = "00000001"`, `ctrl = '1'`

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.

Realice la simulación del banco de pruebas siendo los circuitos a probar los diseños de los Apartados 2.b y 2.c. Incluya en la memoria los dos cronogramas obtenidos al realizar la simulación del banco de pruebas de los dos circuitos diseñado en los Apartados 2.b y 2.c.

Solución al Ejercicio 2

La **entity** del circuito combinacional se muestra en Código VHDL 1.8. El Código VHDL 1.9 muestra el diseño del circuito empleando solo sentencias concurrentes, y los operadores suma y resta. El Código VHDL 1.10 muestra el diseño empleando solo sentencias concurrentes y el operador suma.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity addsub is
5    port( y    : out std_logic_vector(7 downto 0);
6          a, b : in  std_logic_vector(7 downto 0);
7          ctrl : in  std_logic );
8 end addsub;
```

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

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 architecture direct_arq of addsub is
6    signal src0, src1, sum: signed(7 downto 0);
7 begin
8    src0 <= signed(a);
9    src1 <= signed(b);
10   sum  <= src0 + src1  when ctrl='0' else
11           src0 - src1;
12   y    <= std_logic_vector(sum);
13 end direct_arq;
```

Código VHDL 1.9: Apartado 2.b: empleando los operadores suma y resta.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 architecture compartida_arq of addsub is
6    signal src0, src1, sum : signed(8 downto 0);
7    signal b_tmp           : std_logic_vector(7 downto 0);
8    signal cin             : std_logic;   -- bit de acarreo
9 begin
10   src0  <= signed(a & '1');
11   b_tmp <= b when ctrl='0' else not b;
12   src1  <= signed(b_tmp & cin);
13   cin   <= '0' when ctrl='0' else
14            '1';
15   sum   <= src0 + src1;
16   y     <= std_logic_vector(sum(8 downto 1));
17 end compartida_arq;
```

Código VHDL 1.10: Apartado 2.c: empleando solo como operador aritmético la suma.

El banco de pruebas del circuito combinacional se muestra en el Código VHDL 1.11. Los dos cronogramas obtenidos al simular el banco de pruebas, usando como circuitos a testear los diseños de los Apartados 2.b y 2.c, se muestran en las Figuras 1.4 y 1.5.

```vhdl
-- Banco de pruebas
library IEEE;
use IEEE.std_logic_1164.all;
use std.textio.all;

entity bp_addsub is
   constant PERIODO : time       := 10 ns;
end entity bp_addsub;

architecture bp_addsub of bp_addsub is
   signal a         : std_logic_vector(7 downto 0);
   signal b         : std_logic_vector(7 downto 0);
   signal ctrl      : std_logic;
   signal y         : std_logic_vector(7 downto 0);

   component addsub is
      port ( y    : out std_logic_vector(7 downto 0);
             a, b : in  std_logic_vector(7 downto 0);
             ctrl : in  std_logic   );
   end component addsub;

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


   gen_vec_test : process is
      variable error_count: integer := 0;
   begin
      report "Comienza la simulación";

      -- Asigna valor a los vectores de test
      a <= "00000000"; -- Conjunto 1
      b <= "00000000";
      ctrl  <= '0';
      wait for PERIODO;
      if (y /= "00000000") then
         error_count := error_count+1;
         report "ERROR en Conjunto 1. Valor esperado: 00000000";
      end if;    -- if

      ctrl  <= '1';    -- Conjunto 2
      wait for PERIODO;
      if (y /= "00000000") then
         error_count := error_count+1;
         report "ERROR en Conjunto 2. Valor esperado: 00000000";
      end if;    -- if

```

```
50        a <= "00000011"; -- Conjunto 3
51        b <= "00000001";
52        ctrl  <= '0';
53        wait for PERIODO;
54        if (y /= "00000100") then
55             error_count := error_count+1;
56             report "ERROR en Conjunto 3. Valor esperado: 00000100";
57        end if;     -- if
58
59        ctrl  <= '1';    -- Conjunto 4
60        wait for PERIODO;
61        if (y /= "00000010") then
62           error_count := error_count+1;
63           report "ERROR en Conjunto 4. Valor esperado: 00000010";
64        end if;     -- if
65
66        a <= "10001000"; -- Conjunto 5
67        b <= "00010010";
68        ctrl  <= '0';
69        wait for PERIODO;
70        if (y /= "10011010") then
71           error_count := error_count+1;
72           report "ERROR en Conjunto 5. Valor esperado: 10011010";
73        end if;     -- if
74
75        ctrl  <= '1';      -- Conjunto 6
76        wait for PERIODO;
77        if (y /= "01110110") then
78           error_count := error_count+1;
79           report "ERROR en Conjunto 6. Valor esperado: 01110110";
80        end if;     -- if
81
82        a <= "11111111";  -- Conjunto 7
83        b <= "00000001";
84        ctrl  <= '0';
85        wait for PERIODO;
86        if (y /= "00000000") then
87           error_count := error_count+1;
88           report "ERROR en Conjunto 7. Valor esperado: 10011010";
89        end if;     -- if
90
91        ctrl  <= '1';      -- Conjunto 8
92        wait for PERIODO;
93        if (y /= "11111110") then
94           error_count := error_count+1;
95           report "ERROR en Conjunto 8. Valor esperado: 1110110";
96        end if;     -- if
97
98        report "Finaliza la simulación. Hay " &
99               integer'image(error_count)      & " errores";
100       wait;      -- Final del bloque process
101     end process gen_vec_test;
102   end architecture bp_addsub;
```

Código VHDL 1.11: Apartado 2.d: banco de pruebas del circuito.

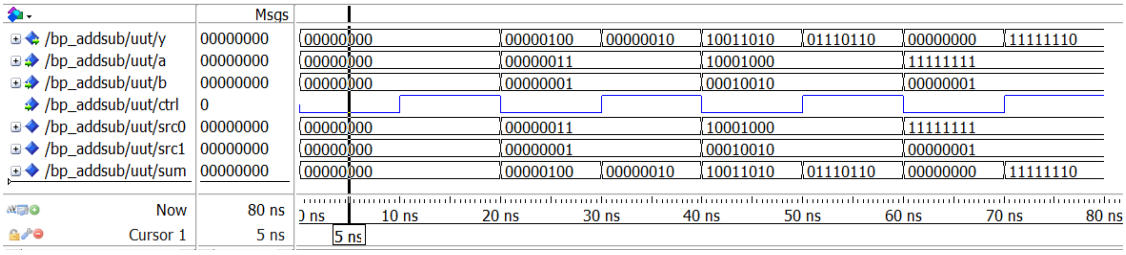

Figura 1.4: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el Apartado 2.b.

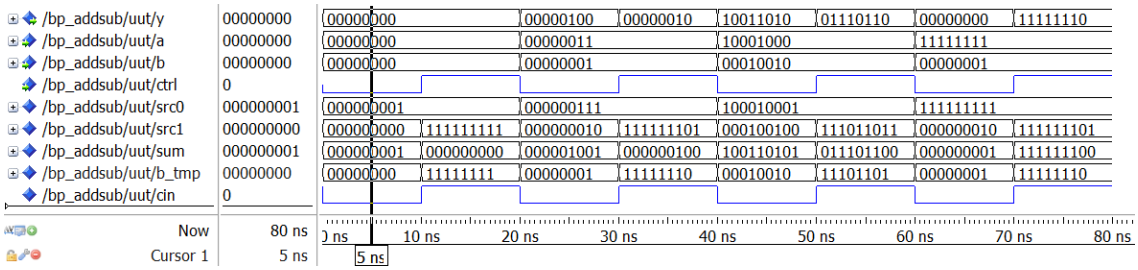

Figura 1.5: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el Apartado 2.c.