

INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Convocatoria ordinaria 2026

Ejercicio 1

Se desea diseñar un circuito digital que implemente la función F mostrada a continuación, que depende de las tres variables x , y y z :

$$F = (x+y') (yz)' (xy'+x'y)$$

- 1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito que implemente la función lógica. Es decir, que tenga tres entradas x , y y z , y una salida F .
- 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 esta función lógica 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 las capturas de pantalla de los dos cronogramas obtenidos al realizar con el simulador de VHDL la simulación del banco de pruebas con los circuitos diseñados en los Apartados 1.b y 1.d.

Solución al Ejercicio 1

La **entity** del circuito solución al Ejercicio 1 se muestra en Código VHDL 1.1.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity funcLog is
5     port ( F      : out std_logic;
6           x, y, z : in  std_logic );
7 end entity funcLog;
```

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

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

```

1 architecture Comp of funcLog is
2     --[ (x+y' ) (yz)' ] (xy' + x'y)
3 begin
4     F <= (x or not y ) and
5         ( not (y and z)) and ((x and not y) or (not x and y));
6 end architecture Comp;
```

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

La Figura 1.1 muestra el diagrama del circuito empleando dos puertas AND de tres entradas, una puerta OR de dos entradas y dos inversores.

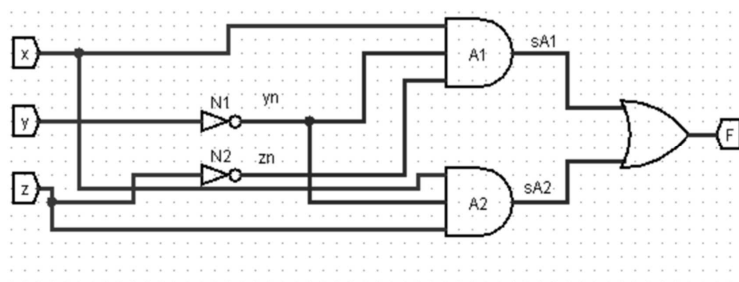


Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

El código VHDL de la **entity** y la **architecture** de las tres puertas lógicas empleadas se muestra en Código VHDL 1.3, 1.4 y 1.5, respectivamente.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity and3 is
5     port ( y0 : out std_logic;
6           x0, x1, x2 : in std_logic );
7 end entity and3;
8 architecture and3 of and3 is
9 begin
10     y0 <= x0 and x1 and x2;
11 end architecture and3;

```

Código VHDL 1.3: Puerta AND de tres entradas.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity or2 is
5     port ( y0 : out std_logic;
6           x0, x1 : in std_logic );
7 end entity or2;
8 architecture or2 of or2 is
9 begin
10     y0 <= x0 or x1;
11 end architecture or2;

```

Código VHDL 1.4: Puerta OR de dos entradas.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity not1 is
5     port ( y0 : out std_logic;
6           x0 : in std_logic );
7 end entity not1;
8 architecture not1 of not1 is
9 begin
10     y0 <= not x0;
11 end architecture not1;

```

Código VHDL 1.5: Puerta NOT con una entrada.

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

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 architecture circuito_Estruc of funcLog is
4 --%=xy'z'+xy'z
5     signal yn, zn, sA1, sA2: std_logic;
6 -- Declaración de las clases de los componentes
7     component and3 is
8         port ( y0 : out std_logic ;
9               x0, x1, x2 : in std_logic);
10    end component and3;
11    component not1 is
12        port ( y0 : out std_logic;
13              x0 : in std_logic );
14    end component not1;
15    component or2 is
16        port ( y0      : out std_logic;
17              x0, x1 : in std_logic );
18    end component or2;
19 begin
20 -- Instanciación y conexión de los componentes
21     N1 : component not1 port map (yn, y);
22     N2 : component not1 port map (zn, z);
23     A1 : component and3 port map (sA1, x, yn, zn);
24     A2 : component and3 port map (sA2, x, yn, z);
25     O1 : component or2 port map (F, sA1, sA2);
26 end architecture circuito_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 dos diseños del circuito combinacional.

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.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 entity bp_funcLog is
6     constant DELAY : time := 20 ns; -- Retardo usado en el test
7 end entity bp_funcLog;
8
9 architecture bp_funcLog of bp_funcLog is
10     signal F : std_logic;
11     signal x, y, z : std_logic;
12
13     component funcLog is
14         port ( F : out std_logic;
15               x, y, z : in std_logic );
16     end component funcLog;
17
18 begin
19     UUT : component funcLog port map
20         (F, x, y, z);
21
22 vec_test : process is
23     variable valor : unsigned (2 downto 0);
24     begin
25         -- Generar todos los posibles valores de entrada
26         for i in 0 to 7 loop
27             valor := to_unsigned(i,3);
28             x <= std_logic(valor(2));
29             y <= std_logic(valor(1));
30             z <= std_logic(valor(0));
31             wait for DELAY;
32         end loop;
33         wait; -- Final de la simulación
34     end process vec_test;
35 end architecture bp_funcLog;

```

Código VHDL 1.7: Apartado 1.e: banco de pruebas.

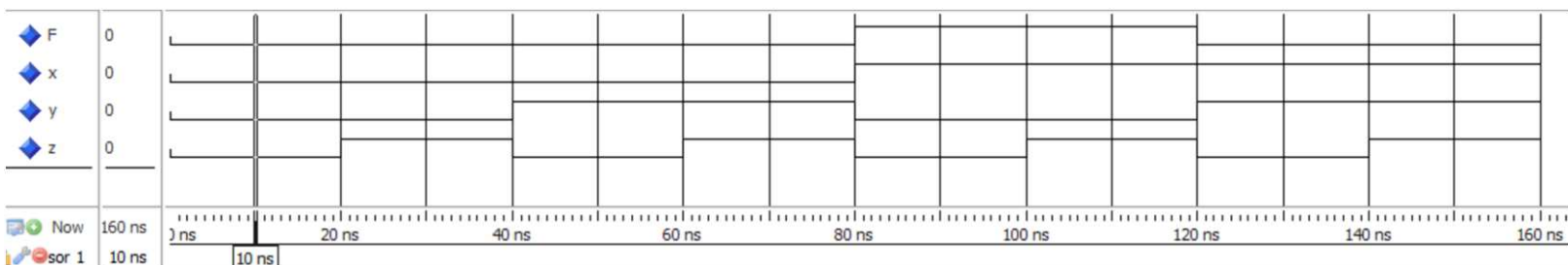


Figura 1.2: Apartado 1.e: simulación del banco de pruebas del diseño describiendo el comportamiento del circuito.

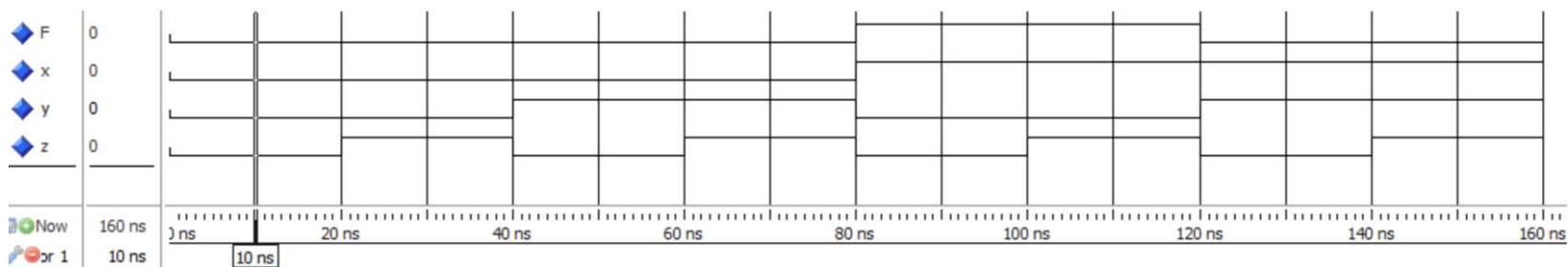


Figura 1.3: Apartado 1.e: simulación del banco de pruebas del diseño describiendo la estructura del circuito.

Ejercicio 2

Se quiere programar en VHDL un circuito combinacional cuya entrada son dos números x e y de 2 bits y cuya salida es un número d de 3 bits cuyo valor representa la diferencia del número x menos el y . Los números x e y se representan en binario sin signo y el número d se representa en binario con signo. El circuito tiene como entrada dos señales x e y de tipo `std_logic_vector` de 2 bits, y como salida la señal d de tipo `std_logic_vector` de 3 bits.

- 2.a) (0.25 puntos) Escriba en VHDL la **entity** del circuito combinacional. Los nombres de los puertos 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) (1.25 puntos) Realice la tabla de verdad del comportamiento del circuito combinacional descrito y a partir de ella obtenga las funciones lógicas que describen el comportamiento de cada bit de la señal de salida. Escriba en la memoria dichas funciones lógicas y muestre también el diagrama de un circuito que implemente las tres 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.
- 2.c) (2.5 puntos) Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.
- 2.d) (2 puntos) Programe en VHDL un banco de pruebas para el circuito combinacional diseñado en el apartado 2.c. El banco de pruebas debe testear todas las posibles entradas al circuito y mostrar al final de la simulación un mensaje con el número total de errores producidos.

Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas del circuito diseñado en el Apartado 2.c.

Solución al Ejercicio 2

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

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity resta is
5     port ( d      : out std_logic_vector(2 downto 0);
6           x, y : in  std_logic_vector(1 downto 0) );
7 end entity resta;
```

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

El diagrama del circuito que implementa las tres funciones lógicas que describen el comportamiento del restador se muestran en la Figura 1.4.

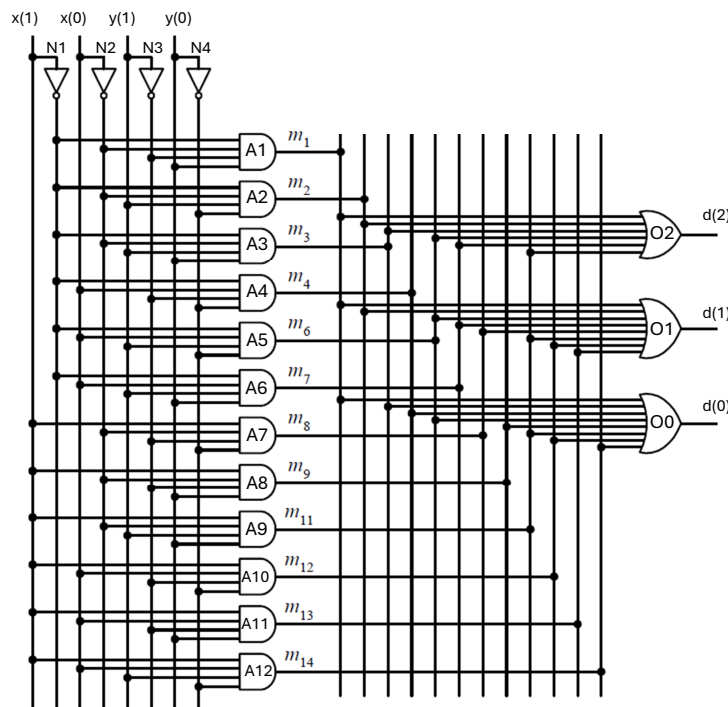


Figura 1.4: Solución al Apartado 2.b: diagrama a nivel de puertas lógicas.

El código VHDL de la **entity** y la **architecture** de las cuatro puertas lógicas empleadas se muestra en Código VHDL 1.9, 1.10, 1.5, y 1.11, respectivamente.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity and4 is
5     port ( y0 : out std_logic;
6           x0, x1, x2, x3 : in  std_logic );
```

```

7 end entity and4;
8 architecture and4 of and4 is
9 begin
10     y0 <= x0 and x1 and x2 and x3;
11 end architecture and4;

```

Código VHDL 1.9: Puerta AND de cuatro entradas.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity or6 is
5     port ( y0 : out std_logic;
6           x0, x1, x2, x3, x4, x5 : in std_logic );
7 end entity or6;
8 architecture or6 of or6 is
9 begin
10     y0 <= x0 or x1 or x2 or x3 or x4 or x5;
11 end architecture or6;

```

Código VHDL 1.10: Puerta OR de seis entradas.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity or8 is
5     port ( y0 : out std_logic;
6           x0, x1, x2, x3, x4, x5, x6, x7 : in std_logic );
7 end entity or8;
8 architecture or8 of or8 is
9 begin
10     y0 <= x0 or x1 or x2 or x3 or x4 or x5 or x6 or x7;
11 end architecture or8;

```

Código VHDL 1.11: Puerta OR de ocho entradas.

La **architecture** describiendo el diseño estructural del circuito restador se muestra en Código VHDL 1.12.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 architecture resta of resta is
5 -- d2 = x1'x0'y1'y0 + x1'x0'y1y0' + x1'x0'y1y0 + x1'x0y1y0'
6 --     + x1'x0y1y0 + x1x0'y1y0
7 -- d1 = x1'x0'y1'y0 + x1'x0'y1y0' + x1'x0y1y0' + x1'x0y1y0
8 --     + x1x0'y1'y0' + x1x0'y1y0 + x1x0y1'y0' + x1x0y1'y0
9 -- d0 = x1'x0'y1'y0 + x1'x0'y1y0 + x1'x0y1'y0' + x1'x0y1y0'
10 --     + x1x0'y1'y0 + x1x0'y1y0 + x1x0y1'y0' + x1x0y1y0'

```

```

11
12  signal xn1, xn0, yn1, yn0, m1, m2, m3, m4, m6,
13      m7, m8, m9, m11, m12, m13, m14: std_logic;
14 -- Declaración de las clases de los componentes
15  component and4 is
16      port ( y0 : out std_logic ;
17            x0, x1, x2, x3 : in std_logic);
18  end component and4;
19  component not1 is
20      port ( y0 : out std_logic;
21            x0 : in std_logic );
22  end component not1;
23  component or6 is
24      port ( y0 : out std_logic;
25            x0, x1, x2, x3, x4, x5 : in std_logic );
26  end component or6;
27  component or8 is
28      port ( y0 : out std_logic;
29            x0, x1, x2, x3, x4, x5, x6, x7 : in std_logic );
30  end component or8;
31  begin
32 -- Instanciación y conexión de los componentes
33  N1 : component not1 port map (xn1, x(1));
34  N2 : component not1 port map (xn0, x(0));
35  N3 : component not1 port map (yn1, y(1));
36  N4 : component not1 port map (yn0, y(0));
37  A1 : component and4 port map (m1, xn1, xn0, yn1, y(0));
38  A2 : component and4 port map (m2, xn1, xn0, y(1), yn0);
39  A3 : component and4 port map (m3, xn1, xn0, y(1), y(0));
40  A4 : component and4 port map (m4, xn1, x(0), yn1, yn0);
41  A5 : component and4 port map (m6, xn1, x(0), y(1), yn0);
42  A6 : component and4 port map (m7, xn1, x(0), y(1), y(0));
43  A7 : component and4 port map (m8, x(1), xn0, yn1, yn0);
44  A8 : component and4 port map (m9, x(1), xn0, yn1, y(0));
45  A9 : component and4 port map (m11, x(1), xn0, y(1), y(0));
46  A10: component and4 port map (m12, x(1), x(0), yn1, yn0);
47  A11: component and4 port map (m13, x(1), x(0), yn1, y(0));
48  A12: component and4 port map (m14, x(1), x(0), y(1), yn0);
49  O0 : component or8 port map (d(0), m1, m3, m4, m6, m9, m11, m12, m14)
50      ;
51  O1 : component or8 port map (d(1), m1, m2, m6, m7, m8, m11, m12, m13)
52      ;
53  O2 : component or6 port map (d(2), m1, m2, m3, m6, m7, m11);
54  end architecture resta;

```

Código VHDL 1.12: Apartado 2.b: architecture del circuito restador.

El banco de pruebas del circuito combinacional restador se muestra en el Código VHDL 1.13. El cronograma del banco de pruebas se muestra en la Figura 1.5.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4
5 entity bp_circ is
6 end entity bp_circ;
7
8 architecture bp_circ of bp_circ is
9     constant WIDTH : integer := 2;
10    constant WIDTHD : integer :=3;
11    --Conectar salidas UUT
12    signal d      : std_logic_vector(WIDTHD-1 downto 0);
13    --Conectar entradas UUT
14    signal x, y   : std_logic_vector(WIDTH-1 downto 0);
15
16    component resta is
17        port ( d      : out std_logic_vector(2 downto 0);
18              x, y : in  std_logic_vector(1 downto 0) );
19    end component resta;
20
21
22 begin
23    -- Instanciar y conectar UUT
24    uut : component resta
25        port map( d, x, y );
26    gen_vec_test : process
27        -- Vectores test
28        variable xv      : unsigned (WIDTH-1 downto 0);
29        variable yv      : unsigned (WIDTH-1 downto 0);
30
31        variable restav : unsigned (WIDTHD-1 downto 0);
32        variable esperado_d : std_logic_vector (WIDTHD-1 downto 0);
33        variable error_count : integer := 0;
34    begin
35        xv := B"00";
36        report "Comienza la simulacion";
37        for count1 in 0 to 2**WIDTH-1 loop
38            yv := B"00";
39            for count2 in 0 to 2**WIDTH-1 loop
40                x <= std_logic_vector(xv);
41                y <= std_logic_vector(yv);
42                restav := ('0' & xv) - ('0' & yv);
43                esperado_d := std_logic_vector(restav);
44                wait for 10 ns;
45                yv := yv + 1;
46                if ( esperado_d /= d ) then
47                    report "ERROR en la salida valida. Valor esperado: "&
48                        std_logic'image(esperado_d(2)) &
49                        std_logic'image(esperado_d(1)) &
50                        std_logic'image(esperado_d(0)) &
51                        ", valor actual: " &
52                        std_logic'image(d(2)) &
53                        std_logic'image(d(1)) &

```

```

54         std_logic'image(d(0)) &
55         " en el instante: " &
56         time'image(now);
57         error_count := error_count + 1;
58     end if;
59 end loop;
60     xv := xv + 1;
61 end loop;
62 report "ERROR: Hay " &
63         integer'image(error_count) &
64         " errores.";
65 wait;
66 end process gen_vec_test;
67 end architecture bp_circ;

```

Código VHDL 1.13: Apartado 2.d: banco de pruebas del circuito restador.

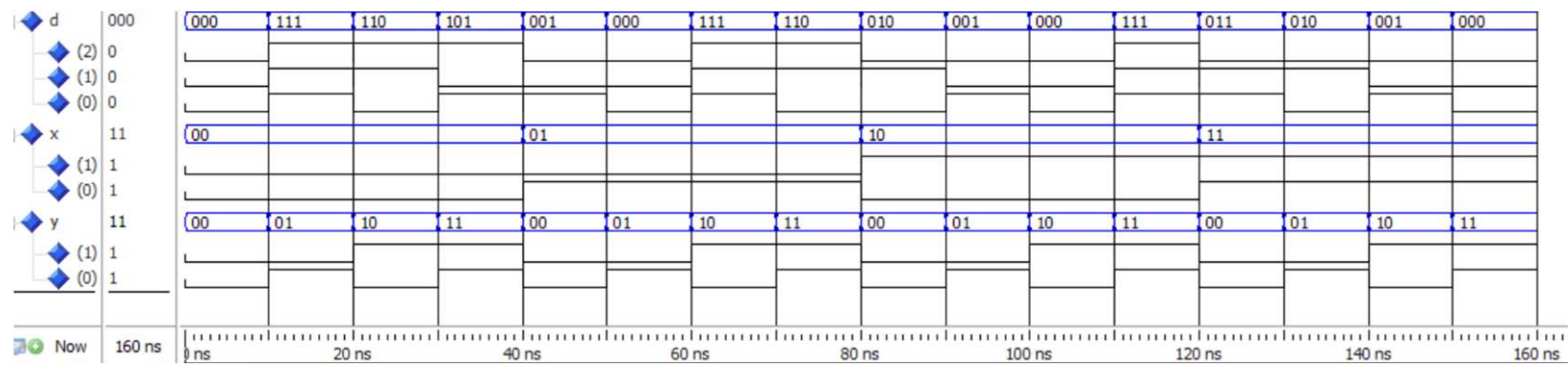


Figura 1.5: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el Apartado 2.c.