INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2020

EJERCICIO 1

Se desea diseñar un circuito digital que implemente las funciones `F1` y `F2` cuya tabla de verdad se muestra a continuación, que dependen de las tres variables `x`, `y` y `z`:

x	y	z	F1	F2
'0'	'0'	'0'	'0'	'0'
'0'	'0'	'1'	'0'	'0'
'0'	'1'	'0'	'0'	'0'
'0'	'1'	'1'	'0'	'1'
'1'	'0'	'0'	'0'	'1'
'1'	'0'	'1'	'1'	'1'
'1'	'1'	'0'	'1'	'1'
'1'	'1'	'1'	'1'	'1'

1.a) (0.5 puntos) Obtenga las funciones lógicas `F1` y `F2` 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 `F1` y `F2`.

1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

1.c) (1 punto) 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) (0.5 puntos) Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, la salida del circuito cuya **entity** ha especificado en el Apartado 1.a. Emplee dicho banco de pruebas para comprobar mediante inspección visual que los dos diseños de los Apartado 1.b y 1.d funcionan correctamente. Incluya en la memoria los dos cronogramas obtenidos al realizar la simulación del banco de pruebas usando en un caso como circuito de test el circuito de Apartado 1.b y en el otro caso el circuito del Apartado 1.d.

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 funcF1F2 is
   port ( F1, F2: out std_logic;
               x, y, z: in std_logic );
end entity funcF1F2;
-------------------------------------
```

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

La Figura 1.1 muestra el diagrama del circuito implementado empleando dos puertas AND de dos entradas y dos puertas OR de dos entradas.

El código VHDL de la **entity** y la **architecture** de estas dos puertas lógicas se muestra en Código VHDL 1.3 y 1.4, respectivamente. El Código VHDL 1.5 muestra la **architecture** del circuito describiendo su estructura.

```
----------------------------------
--F1 = x(y+z)
--F2 = x+yz
architecture Comp of funcF1F2 is
begin
  F1 <= x and (y or z);
  F2 <= x or (y and z);
end architecture Comp;
----------------------------------
```

Código VHDL 1.2: Solución al Apartado 1.b: **architecture** del circuito describiendo su comportamiento.

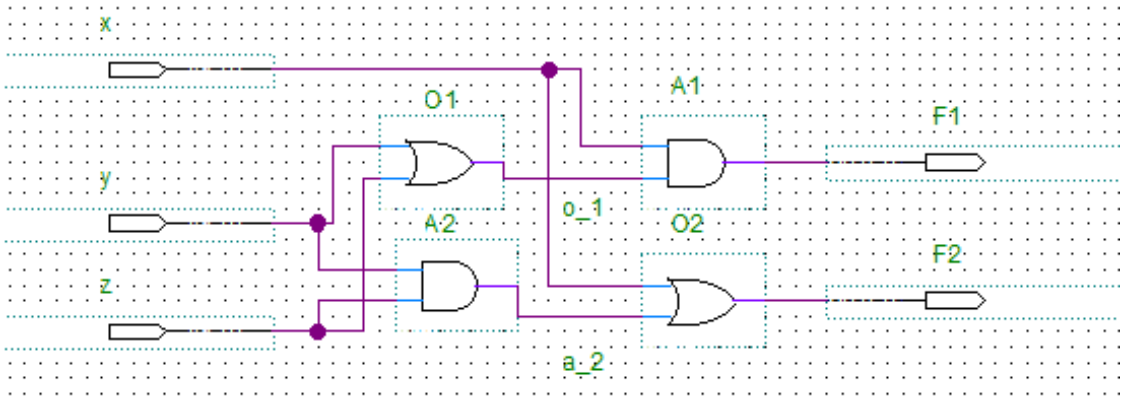

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

```
----------------------------------
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.

```
-------------------------------------
--F1 = x(y+z)
--F2 = x+yz
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcF1F2 is
   signal a_2: std_logic;
   signal o_1: 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 or2 is
      port ( y0 : out std_logic;
           x0, x1 : in std_logic );
   end component or2;
begin
-- Instanciación y conexión de los componentes
   O1 : component or2 port map (o_1, y, z);
   A1 : component and2 port map (F1, x, o_1);
   A2 : component and2 port map (a_2, y, z);
   O2 : component or2 port map (F2, x, a_2);
end architecture circuito_Estruc;
-----------------------------------
```

Código VHDL 1.5: 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 1.6. Los dos cronogramas obtenidos al simular el banco de pruebas se muestran en las Figuras 1.2–1.3.

```
-- Banco de pruebas del circuito Ejercicio 1
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_funcF1F2 is
    constant DELAY      : time      := 20 ns; -- Retardo usado en el test
end entity bp_funcF1F2;

architecture bp_funcF1F2 of bp_funcF1F2 is
    signal F1, F2: std_logic;
    signal x, y, z: std_logic;

    component funcF1F2 is
      port (   F1, F2 : out std_logic;
               x, y, z: in std_logic );
      end component funcF1F2;

begin
    UUT : component funcF1F2 port map
          (F1, F2, 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_funcF1F2;
-------------------------------------------------
```

Código VHDL 1.6: Solución al Apartado 1.e: banco de pruebas del circuito.

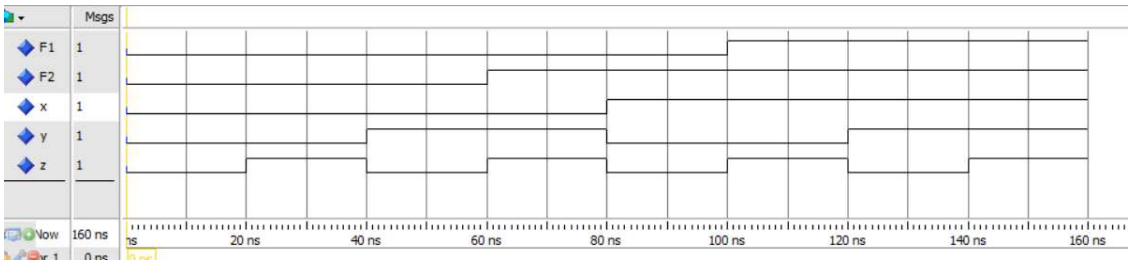

Figura 1.2: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.b.

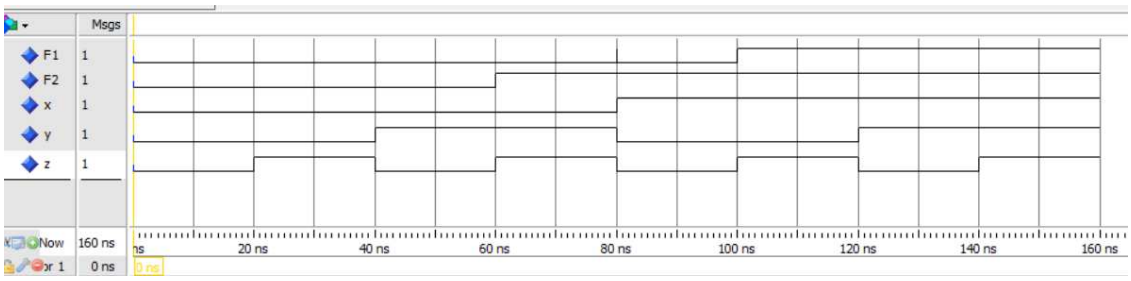

Figura 1.3: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.d.

EJERCICIO 2

Se quiere programar en VHDL un circuito combinacional que tiene una señal de entrada de 2^{n-1} bits llamada `X` y dos señales de salida, la señal de 1 bit llamada `ERROR` y la señal de n bits llamada `CUENTA`.

El circuito ha de comportarse del modo siguiente. Determina si la señal de entrada `X` no tiene ningún cero ó tiene una única serie consecutiva de ceros. En ese caso, la señal `ERROR` ha de tener valor '0' y la señal `CUENTA` tiene como valor el número de ceros, en representación binaria sin signo, de la única secuencia de ceros encontrada en la señal de entrada. Por el contrario, si hay más de una secuencia de ceros, el valor de la señal `ERROR` es '1' y la señal `CUENTA` tiene el valor cero en binario.

En la siguiente tabla se pueden ver algunos valores del circuito para el caso en que $n = 4$.

X	ERROR	CUENTA
"00000000"	'0'	"1000"
"00111000"	'1'	"0000"
"11000111"	'0'	"0011"
"11111111"	'0'	"0000"

2.a) (2 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento del circuito combinacional empleando sólo un bloque **process** y sentencias secuenciales. 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. Se ha de declarar n como una constante de tipo **generic**. El diseño ha de ser válido para cualquier valor de n.

2.b) (1 punto) Deduzca las tablas de verdad correspondientes a las señales de entrada y salida del circuito para un valor de $n = 3$. Entonces, obtenga las funciones lógicas correspondientes y escriba en VHDL la **architecture** del circuito combinacional que describe el funcionamiento del circuito empleando dichas funciones lógicas.

Dibuje el diagrama de un circuito que implemente las señales de salida empleando 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) (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.

2.d) (2 puntos) Programe en VHDL un banco de pruebas que testee el circuito diseñado en los Apartados 2.a, 2.b y 2.c para el caso en que el valor de n sea 3. Compruebe el funcionamiento del circuito únicamente para los siguientes valores de la señal de entrada: "0000", "0001", "0110", "1010", "1110" y "1111". 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.

Incluya en la memoria los cronogramas obtenidos al realizar la simulación del banco de pruebas de los circuitos diseñados en los Apartados 2.a, 2.b y 2.c.

Solución al Ejercicio 2

La **entity** del circuito combinacional se muestra en Código VHDL 1.7.

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

entity ContadorGen is
generic (n:integer:=3);
port(CUENTA: out std_logic_vector(n-1  downto 0);
        ERROR: out std_logic;
              X: in std_logic_vector(2**(n-1)-1  downto    0));
end;
----------------------------------
```

Código VHDL 1.7: Solución al Apartado 2.a: **entity** del circuito.

El Código VHDL 1.8 muestra el código VHDL del circuito combinacional empleando sólo un bloque **process** y sentencias secuenciales.

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

architecture ContadorGenAlg of ContadorGen is
begin
   process(X)
      variable TEMP_CUENTA : INTEGER range 0 to 2**(n-1);
      variable VISTO_CERO, VISTO_SECUENCIA : BOOLEAN;
   begin
      ERROR <= '0';
      VISTO_CERO := FALSE;
      VISTO_SECUENCIA := FALSE;
      TEMP_CUENTA := 0;
      for I in 0 to 2**(n-1)-1 loop
            if (VISTO_SECUENCIA and X(I) = '0') then
                  TEMP_CUENTA := 0;
                  ERROR <= '1';
                  exit;
            elsif (VISTO_CERO and X(I) = '1') then
                  VISTO_SECUENCIA := TRUE;
            elsif (X(I) = '0') then
                  VISTO_CERO := TRUE;
                  TEMP_CUENTA := TEMP_CUENTA + 1;
            end if;
      end loop;
      CUENTA <= std_logic_vector(to_unsigned(TEMP_CUENTA,n));
   end process;
end ContadorGenAlg;
```

Código VHDL 1.8: Solución al Apartado 2.a: circuito combinacional descrito usando un bloque **process**.

El Código VHDL 1.9 muestra el código VHDL del circuito combinacional descrito con las funciones lógicas deducidas a partir de las tablas de verdad.

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

architecture ContadorGenFunLog of ContadorGen is
   signal xn3, xn1, xn2, xn0 : std_logic;
begin
    xn3 <= not x(3);
    xn2 <= not x(2);
    xn1 <= not x(1);
    xn0 <= not x(0);
    ERROR <= (xn3 and x(2) and xn1) or (xn2 and x(1) and xn0) or
             (xn3 and x(2) and xn0) or (xn3 and x(1) and xn0);
    CUENTA(2) <= xn3 and xn2 and xn1 and xn0;
    CUENTA(1) <= (xn3 and xn2 and x(0)) or (x(3) and xn1 and xn0) or
                 (x(3) and   xn2 and xn1);
    CUENTA(0) <= (xn3 and xn2 and xn1 and x(0)) or
                 (xn3 and x(2) and x(1) and x(0)) or
                 (x(3) and xn2 and xn1 and xn0) or
                 (x(3) and xn2 and x(1) and x(0)) or
                 (x(3) and x(2) and xn1 and x(0)) or
                 (x(3) and x(2) and x(1) and xn0);
end ContadorGenFunLog;
```

Código VHDL 1.9: Solución al Apartado 2.b: circuito combinacional descrito mediante funciones lógicas.

La Figura 1.4 muestra el diagrama al nivel de puertas lógicas. El código VHDL de la **entity** y la **architecture** de las puertas lógicas OR de dos entradas, inversor y la puerta AND de tres y de cuatro entradas se muestra en Código VHDL 1.4, 1.10, 1.11 y 1.12, respectivamente. Los códigos VHDL 1.13 y 1.14 muestran la **architecture** del circuito describiendo su estructura.

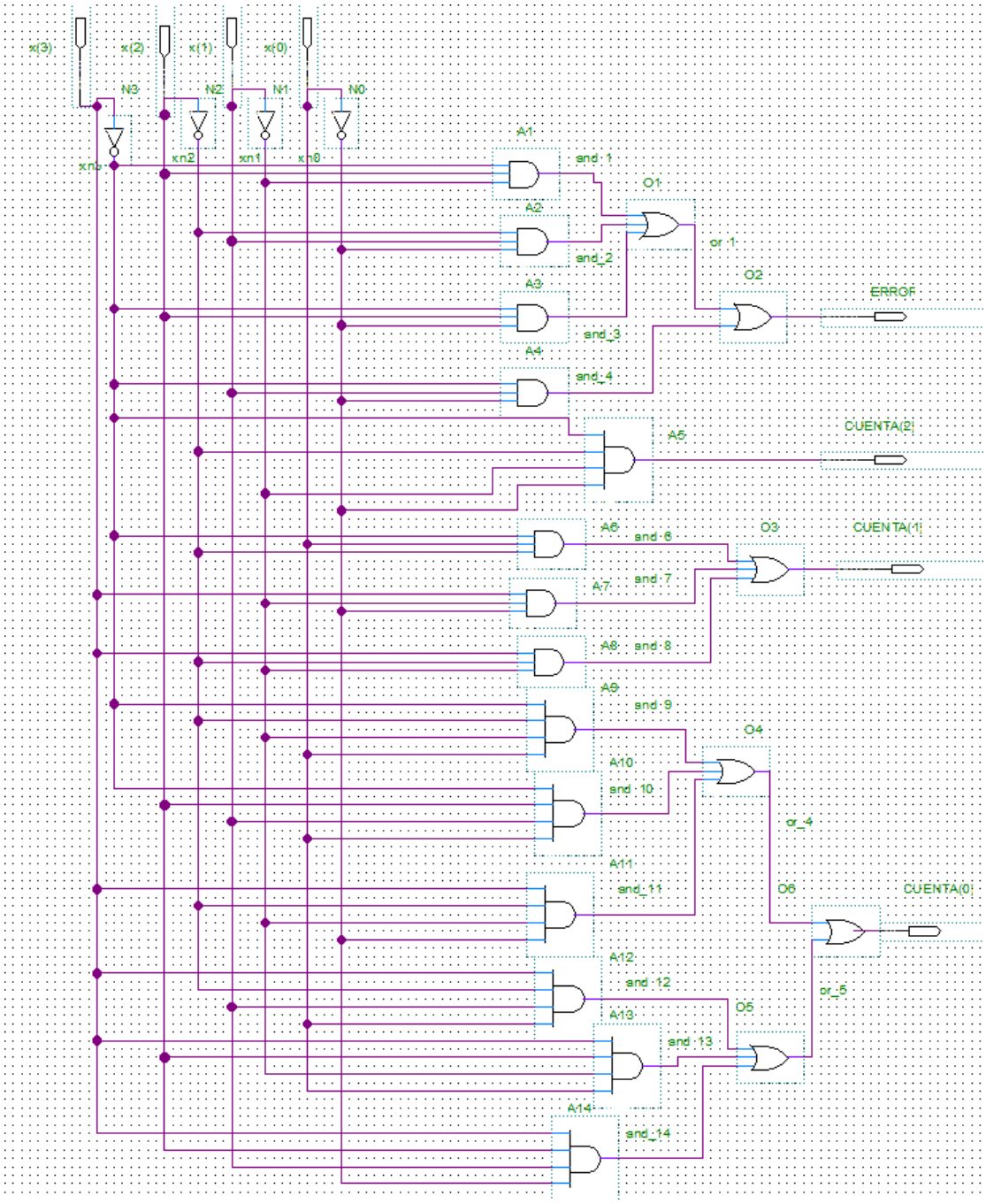

Figura 1.4: Solución al Apartado 2.b: diagrama al nivel de puertas lógicas.

```
----------------------------------
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 1.10: Puerta NOT lógica.

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

entity and3 is
   port ( y0 : out std_logic;
        x0, x1, x2 : in std_logic );
end entity and3;

architecture and3 of and3 is
begin
   y0 <= x0 and x1 and x2;
end architecture and3;
----------------------------------
```

Código VHDL 1.11: Puerta lógica AND de 3 entradas.

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

entity and4 is
   port ( y0 : out std_logic;
        x0, x1, x2, x3 : in std_logic );
end entity and4;

architecture and4 of and4 is
begin
   y0 <= x0 and x1 and x2 and x3;
end architecture and4;
----------------------------------
```

Código VHDL 1.12: Puerta lógica AND de 4 entradas.

El banco de pruebas del circuito combinacional se muestra en Código VHDL 1.15–1.16. Los cronogramas obtenidos al simular el banco de pruebas usando como

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

architecture ContadorCirc of ContadorGen is
   signal xn3, xn1, xn2, xn0 : std_logic;
   signal and_1, and_2, and_3, and_4, and_5: std_logic;
   signal and_6, and_7, and_8, and_9, and_10: std_logic;
   signal and_11, and_12, and_13, and_14: std_logic;
   signal or_1, or_4, or_5: std_logic;
-- Declaración de las clases de los componentes
   component and4 is
      port (   y0 : out std_logic ;
            x0, x1, x2, x3 : in std_logic);
   end component and4;
   component and3 is
      port (   y0 : out std_logic ;
            x0, x1, x2 : in std_logic);
   end component and3;
   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;
   component or3 is
      port ( y0 : out std_logic;
           x0, x1, x2 : in std_logic );
   end component or3;
begin
```

Código VHDL 1.13: Solución al Apartado 2.c: descripción estructural del circuito.

circuito a testear los diseños de los apartados 2.a, 2.b y 2.c se muestran, respectivamente, en las Figuras 1.5, 1.6 y 1.7.

```
-- Instanciación y conexión de los componentes
   N0 : component not1  port map (xn0, x(0));
   N1 : component not1  port map (xn1, x(1));
   N2 : component not1  port map (xn2, x(2));
   N3 : component not1  port map (xn3, x(3));
   A1: component and3  port map (and_1, xn3, x(2), xn1);
   A2: component and3  port map (and_2, xn2, x(1), xn0);
   A3: component and3  port map ( and_3, xn3, x(2), xn0);
   A4: component and3  port map (and_4, xn3, x(1), xn0);
   O1: component or3  port map (or_1, and_1, and_2, and_3);
   O2: component or2  port map( ERROR, or_1, and_4);
   A5: component and4  port map(CUENTA(2), xn3, xn2, xn1, xn0);
   A6: component and3  port map(and_6, xn3, xn2, x(0));
   A7: component and3  port map(and_7, x(3), xn1, xn0);
   A8: component and3  port map(and_8, x(3), xn2, xn1);
   O3: component or3  port map(CUENTA(1), and_6, and_7, and_8);
   A9: component and4  port map(and_9, xn3, xn2, xn1, x(0));
   A10: component and4  port map(and_10, xn3, x(2), x(1), x(0));
   A11: component and4  port map(and_11, x(3), xn2, xn1, xn0);
   A12: component and4  port map(and_12, x(3), xn2, x(1), x(0));
   A13: component and4  port map (and_13, x(3), x(2), xn1, x(0));
   A14: component and4  port map (and_14, x(3), x(2), x(1), xn0);
   O4: component or3  port map(or_4,and_9,and_10,and_11);
   O5: component or3  port map(or_5,and_12, and_13, and_14);
   O6: component or2  port map(CUENTA(0), or_4, or_5);
   end ContadorCirc;
```

Código VHDL 1.14: Solución al Apartado 2.c: continuación de la descripción estructural del circuito.

```
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 CUENTA      : std_logic_vector(2 downto 0);   -- Conectar salidas UUT
   signal ERROR: std_logic;
   signal X   : std_logic_vector(3 downto 0);   -- Conectar entradas UUT

component ContadorGen is
   generic (n:integer:=3);
   port(CUENTA: out std_logic_vector(n-1 downto 0);
        ERROR: out std_logic;
             X: in std_logic_vector(2**(n-1)-1 downto   0));
end component ContadorGen;
begin
   -- Instanciar y conectar UUT
   uut : component ContadorGen
         port map( CUENTA, ERROR, X );
   gen_vec_test : process
      variable tx      : unsigned (3 downto 0); -- Vector de test
      variable esperado_Y     : std_logic_vector(4 downto 0);
      variable error_count          : integer     := 0;
   begin
      report "Comienza la simulación";

       X <= "0000";
       wait for 10 ns;
       if (ERROR/='0' or CUENTA /= "100") then
            report "ERROR en la entrada 0000";
       end if;

       X <= "0001";
       wait for 10 ns;
       if (ERROR/='0' or CUENTA /= "011") then
            report "ERROR en la entrada 0001";
       end if;

       X <= "0110";
       wait for 10 ns;
       if (ERROR/='1' or CUENTA /= "000") then
            report "ERROR en la entrada 0110";
       end if;
```

Código VHDL 1.15: Solución al Apartado 2.d: banco de pruebas del circuito.

```
        X <= "1010";
        wait for 10 ns;
        if (ERROR/='1' or CUENTA /= "000") then
            report "ERROR en la entrada 1010";
        end if;

        X <= "1110";
        wait for 10 ns;
        if (ERROR/='0' or CUENTA /= "001") then
            report "ERROR en la entrada 1110";
        end if;

        X <= "1111";
        wait for 10 ns;
        if (ERROR/='0' or CUENTA /= "000") then
            report "ERROR en la entrada 1111";
        end if;

      report  "ERROR: Hay "                &
                integer'image(error_count) &
                " errores.";
      wait;
   end process gen_vec_test;
end architecture bp_circ;
```

Código VHDL 1.16: Solución al Apartado 2.d: continuación del banco de pruebas del circuito.

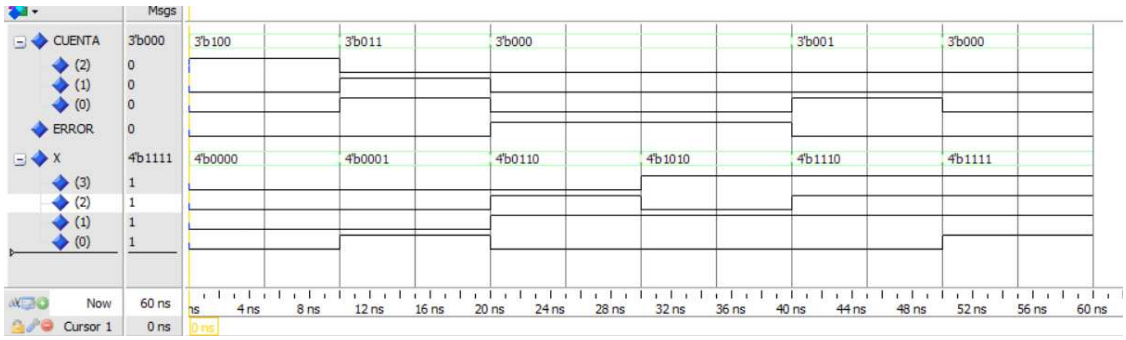

Figura 1.5: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.a.

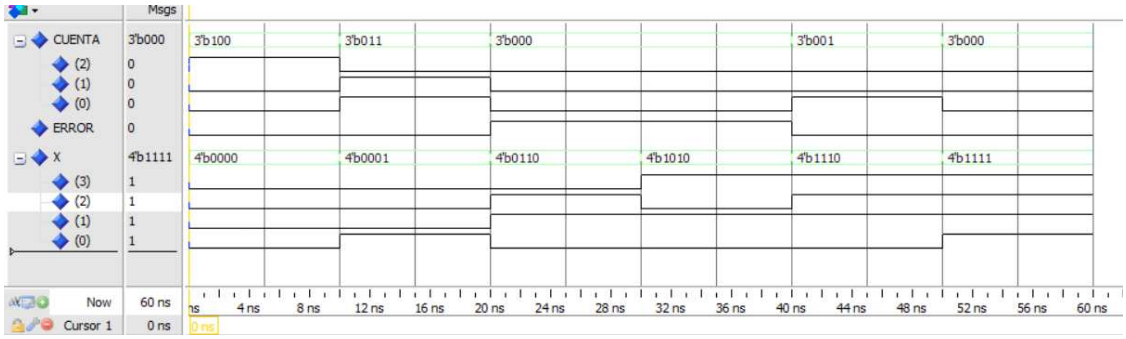

Figura 1.6: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.b.

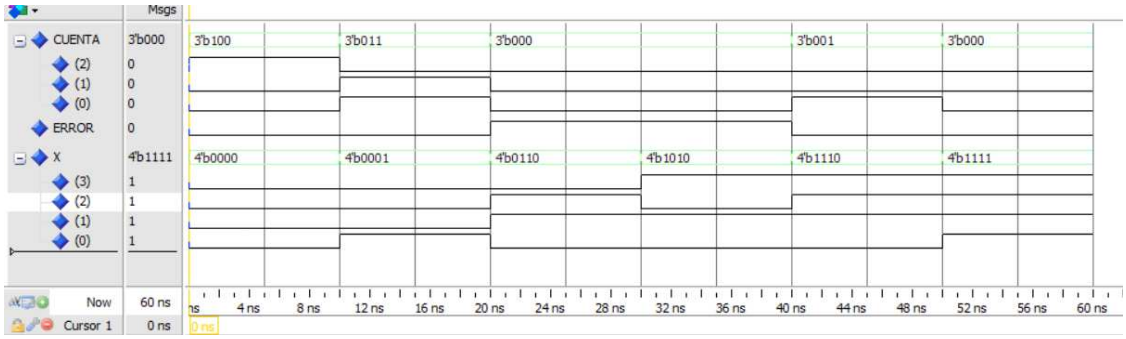

Figura 1.7: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.c.