INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Convocatoria ordinaria 2025

Ejercicio 1

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

```
F = xy'z' + x' + xyz'
G = xy + x'z + yz
```

1.a) (0.5 puntos) 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 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, G : 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 begin
3  F <= (x and not  y and not z) or (not x) or
4       (x and y and not z);
5  G <= (x and y) or ( not x and z) or
6       (y and z);
7 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 dos entradas, dos puertas OR de dos entradas y dos inversores.

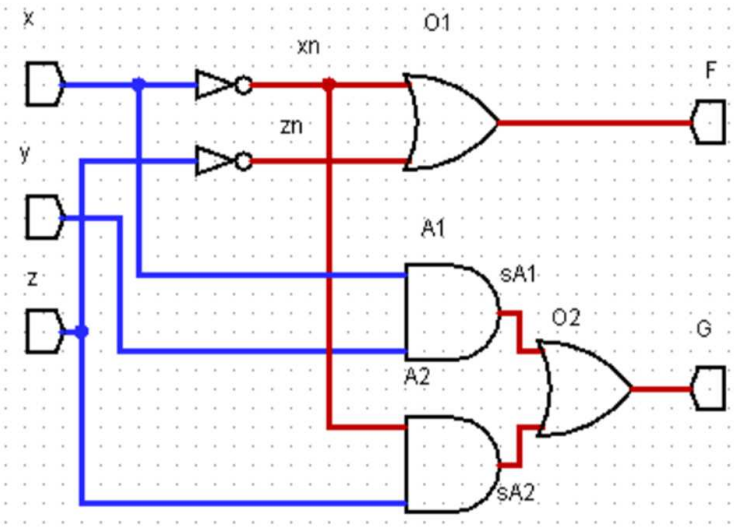

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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

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

entity and2 is
  port ( y0 : out std_logic;
     x0, x1 : in std_logic );
end entity ;
architecture and2  of and2 is
begin
  y0 <= x0 and x1;
end architecture and2;
```

Código VHDL 1.3: 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: 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: Puerta NOT con una entrada.

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

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcLog is
  signal xn, zn, sA1, sA2: 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 (zn, z);
  A1 : component and2 port map (sA1, x, y);
  A2 : component and2 port map (sA2, xn, z);
  O1 : component or2 port map (F, xn, zn);
  O2 : component or2 port map (G, sA1, sA2);
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, G : std_logic;
11    signal x, y, z  : std_logic;
12
13    component funcLog is
14     port (  F, G : 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, G, 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.

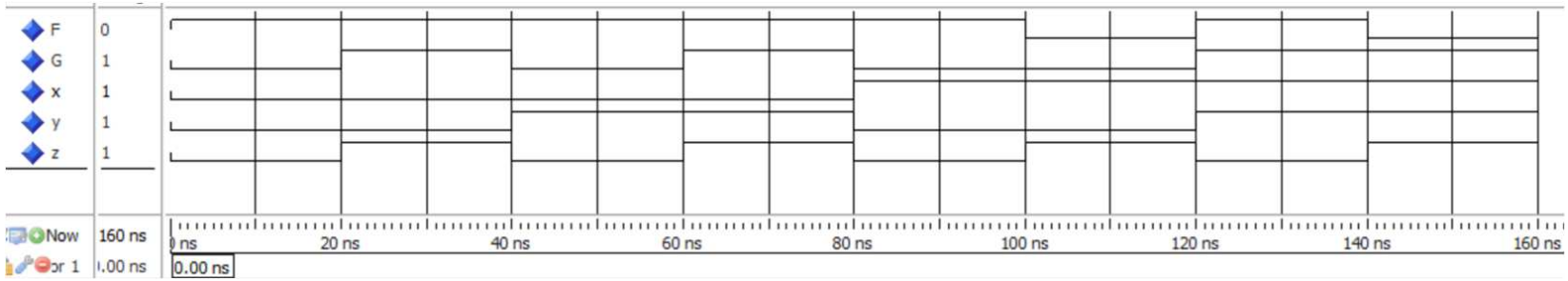

Figura 1.2: Apartado 1.e: simulación del banco de pruebas del diseño describiendo el comportamiento del circuito.

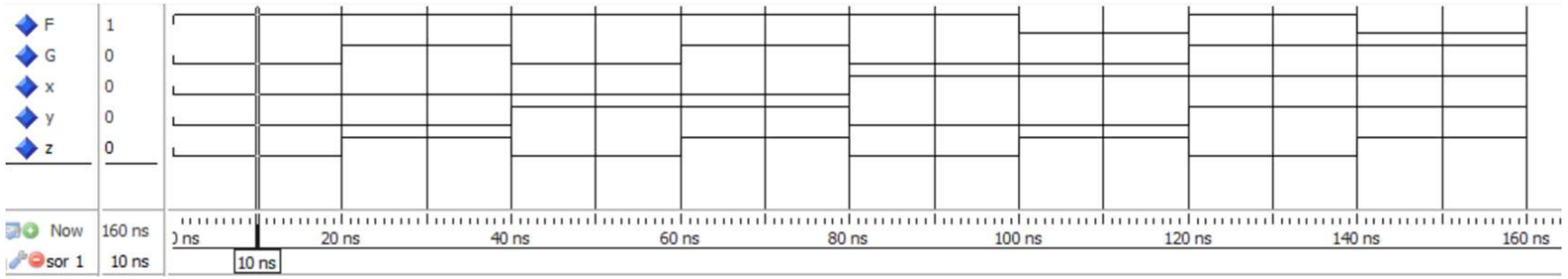

Figura 1.3: Apartado 1.e: simulación del banco de pruebas del diseño describiendo la estructura del circuito.

Ejercicio 2

Se quiere programar en VHDL un circuito combinacional para comparar dos números x e y de N bits, siendo N mayor o igual que uno. Los números x e y se representan en binario sin signo. El circuito tiene como señales de entrada dos números de tipo `std_logic_vector` de `N` bits, siendo `N` una constante de tipo `generic` cuyo valor ha de ser mayor o igual que 1. El circuito tiene dos señales de salida llamadas `gout` y `eout`, que nos indican si x es mayor que y o ambos números son iguales. La señal `gout` tiene valor `1` sólo si `x` es mayor que `y`, teniendo valor `0` en caso contrario. La señal `eout` tiene valor `1` sólo si `x` es igual que `y`, teniendo valor `0` en caso contrario.

Este circuito se va a construir de manera iterativa. En la siguiente figura se muestra el módulo cuando se comparan dos números de 1 bit y la estructura del circuito comparador de 4 bits empleando 4 módulos comparadores de 1 bit.

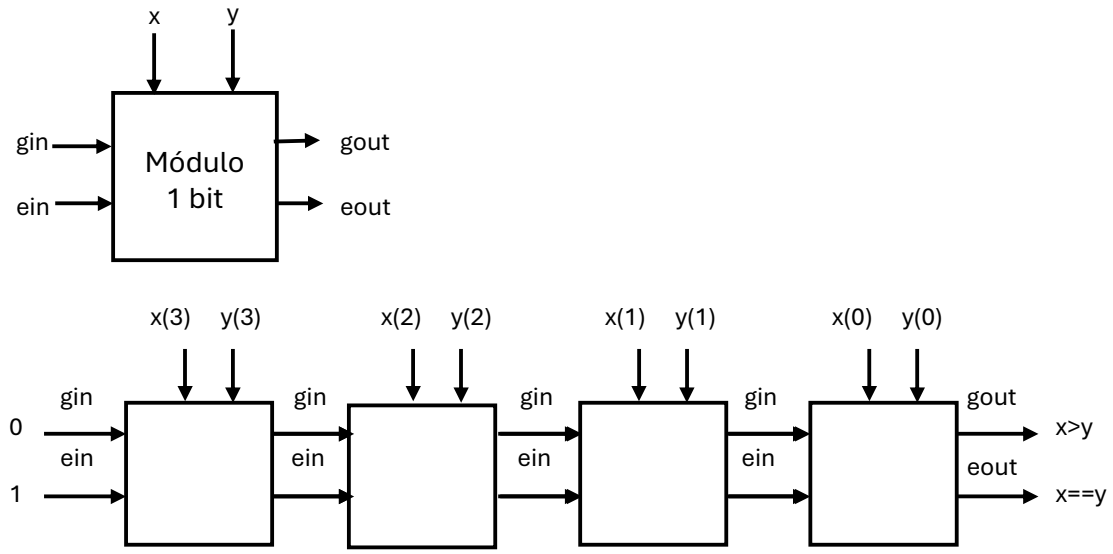

Figura 1.4: Comparador de 1 bit y comparador de 4 bits usando 4 comparadores de 1 bit.

2.a) (0.25 puntos) Escriba en VHDL la **entity** del circuito combinacional que compara dos números de un bit y cuyo módulo se dió en la anterior figura, así como la **entity** del circuito combinacional capaz de comparar dos números de N bits como se ha descrito anteriormente. El número de bits de las señales de entrada tiene que ser una constante de tipo `generic`.

En ambas **entity**, 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) A partir de la tabla de verdad del comportamiento del módulo que compara dos números de un bit obtenga las funciones lógicas que describen el comportamiento de cada señal de salida. 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. 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.c) (2.5 puntos) Escriba en VHDL la **architecture** que describe la estructura del circuito combinacional comparador de N bits, donde N es una contante de tipo `generic`, usando el comparador de 1 bit desarrollado en el anterior apartado. Emplee sentencias generate para realizar la instanciación y conexión de los N comparadores. El diseño debe ser válido para cualquier valor de N que sea mayor o igual a 1.

2.d) (2 puntos) Programe en VHDL un banco de pruebas para el comparador de N bits, que sea válido para cualquier valor de N. 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.

Realice la simulación del banco de pruebas para un valor de $N = 4$ siendo el circuito a probar el diseño del Apartado 2.c cuando el valor de N es 4. 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 comparador de 1 bit y del circuito comparador de N bits se muestran en Código VHDL 1.8–1.9.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity compara1bit is
5   port ( gout, eout : out std_logic;
6          x, y, gin, ein : in std_logic );
7 end entity compara1bit;
```

Código VHDL 1.8: Apartado 2.a: **entity** del circuito comparador de 1 bit.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity comparaNbits is
5   generic( N: integer := 4);
6   port    ( gout, eout    : out std_logic;
7             x, y  : in std_logic_vector(N-1 downto 0));
8 end entity comparaNbits;
```

Código VHDL 1.9: Apartado 2.a: **entity** del circuito comparador de N bits.

El diagrama del circuito que implementa las dos funciones lógicas que describen el comportamiento del comparador de 1 bit se muestran en la Figura 1.5.

El código VHDL de la **entity** y la **architecture** de las cinco puertas lógicas empleadas se muestra en Código VHDL 1.3, 1.4, 1.5, 1.10 y 1.11, 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 1.10: Puerta AND de tres entradas.

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity or3 is
```

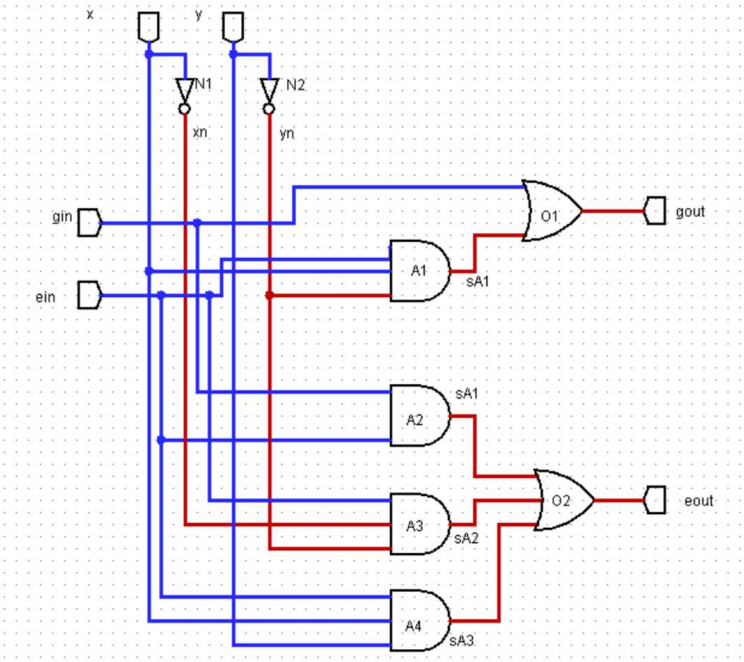

Figura 1.5: Solución al Apartado 2.b: diagrama a nivel de puertas lógicas.

```
  port ( y0 : out std_logic;
     x0, x1, x2 : in std_logic );
end entity or3;
architecture or3 of or3 is
begin
  y0 <= x0 or x1 or x2;
end architecture or3;
```

Código VHDL 1.11: Puerta OR de tres entradas.

La **architecture** describiendo el diseño estructural del circuito comparador de 1 bit se muestra en Código VHDL 1.12.

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture compara1bit of compara1bit is
  signal xn, yn, sA1, sA2, sA3, sA4: 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 and3 is
    port ( y0          : out std_logic ;
           x0, x1, x2 : in std_logic);
  end component and3;
  component not1 is
    port ( y0 : out std_logic;
```

```
16            x0 : in std_logic );
17    end component not1;
18    component or2 is
19      port ( y0     : out std_logic;
20             x0, x1 : in std_logic );
21    end component or2;
22    component or3 is
23      port ( y0 : out std_logic;
24             x0, x1, x2 : in std_logic );
25    end component or3;
26  begin
27  -- Instanciación y conexión de los componentes
28    N1 : component not1 port map (xn, x);
29    N2 : component not1 port map (yn, y);
30    A1 : component and3 port map (sA1, ein, x, yn);
31    A2 : component and2 port map (sA2, gin, ein);
32    A3 : component and3 port map (sA3, ein, xn, yn);
33    A4 : component and3 port map (sA4, ein, x, y);
34    O1 : component or2 port map (gout, gin, sA1);
35    O2 : component or3 port map (eout, sA2, sA3, sA4);
36  end architecture compara1bit;
```

Código VHDL 1.12: Apartado 2.b: **architecture** del circuito comparador de 1 bits.

La **architecture** describiendo el diseño estructural del circuito comparador de N bits se muestra en Código VHDL 1.13.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3
4  architecture comparaNbits of comparaNbits is
5     signal pgout: std_logic_vector(N downto 0);
6     signal peout: std_logic_vector(N downto 0);
7    -- signal gout, eout: std_logic;
8     component compara1bit is
9       port ( gout, eout     : out std_logic;
10             x, y, gin, ein : in std_logic);
11    end component compara1bit;
12
13 begin
14    pgout(N) <= '0';
15    peout(N) <= '1';
16 --Instanciación y conexión de componentes usando GENERATE
17    conexion: for k in N-1 downto 0 generate --GENERATE iterativo
18        comparadorNbits: compara1bit
19        port map (pgout(k), peout(k), x(k), y(k),
20                  pgout(k+1), peout(k+1));
21    end generate conexion;
22    gout <= pgout(0);
23    eout <= peout (0);
24 end architecture comparaNbits;
```

Código VHDL 1.13: Apartado 2.c: **architecture** del circuito comparador de N bits.

El banco de pruebas del circuito combinacional comparador de N bits se muestra en el Código VHDL 1.14. El cronograma del banco de pruebas se muestra en la Figura 1.6.

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

entity bp_comparaNbits is
end entity bp_comparaNbits;

architecture bp_comparaNbits of bp_comparaNbits is
  constant Nbits : integer := 4;
  signal gout, eout : std_logic;  -- Conectar salidas UUT
  signal x, y : std_logic_vector(Nbits-1 downto 0);  -- Conectar
   entradas UUT

  component comparaNbits is
    generic( N: integer := Nbits);
    port   ( gout, eout : out std_logic;
             x, y       : in std_logic_vector(N-1 downto 0));
  end component comparaNbits;

begin
  -- Instanciar y conectar UUT
  uut : component comparaNbits port map
        ( gout => gout, eout => eout, x => x, y => y );
  gen_vec_test : process
    variable num_errores  : integer := 0;          -- Numero de errores
  begin
    for op_X in 0 to 2**Nbits-1 loop
      for op_Y in 0 to 2**Nbits-1 loop
          x    <= std_logic_vector(to_unsigned(op_x,Nbits));
          y    <= std_logic_vector(to_unsigned(op_y,Nbits));
          wait for 10 ns;
          -- Comprueba resultado
          if (x/=y and eout='1') or (x=y and eout='0') then
            report "Error. Valor eout: " &
                   std_logic'image(eout)&
                   " Valores operandos (x,y): "&
                   integer'image(op_x) &
                   ", " &
                   integer'image(op_y);
            num_errores := num_errores + 1;
         elsif (x>y and gout ='0') or (x<y and gout ='1') then
            report "Error. Valor gout: " &
                   std_logic'image(gout)&
                   " Valores operandos (x, y): "&
                   integer'image(op_x) &
                   ", " &
                   integer'image(op_y);
            num_errores := num_errores + 1;
          end if;
      end loop;
    end loop;
```

```
51
52      report "Test completo. Hay "       &
53             integer'image(num_errores)  &
54             " errores.";
55      wait; --Final simulación
56    end process gen_vec_test;
57  end architecture bp_comparaNbits;
```

Código VHDL 1.14: Apartado 2.d: banco de pruebas del circuito comparador de N bits.

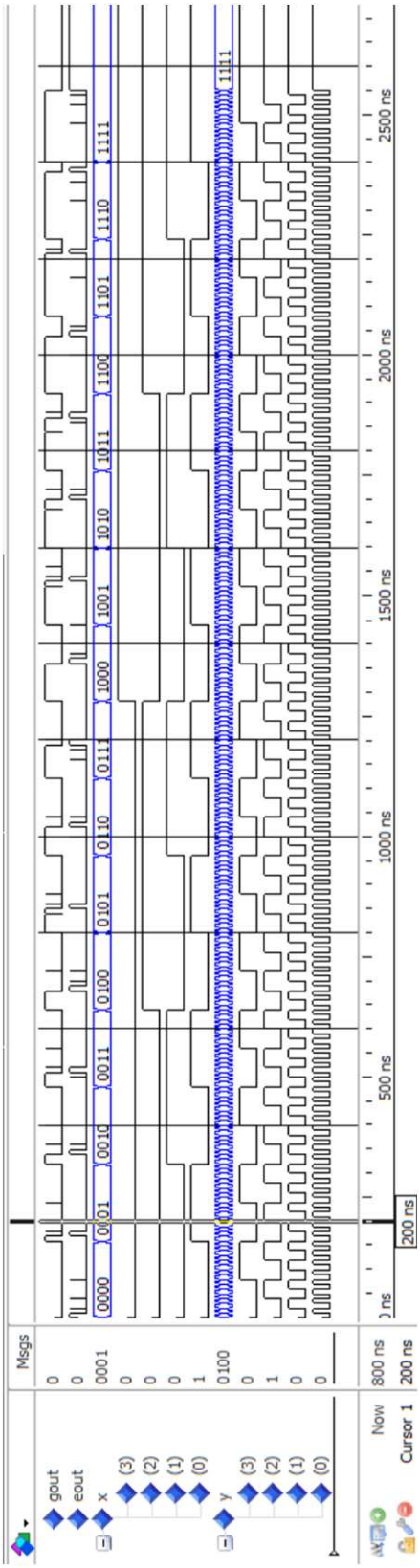

Figura 1.6: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el Apartado 2.c.