INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Convocatoria ordinaria 2024

Ejercicio 1

En la Figura 1.1 se muestra el símbolo lógico de un circuito digital cuya función es contabilizar el número de señales de entrada que tienen valor par. Tanto las señales de entrada como de salida del circuito se interpretan como números binarios sin signo. Si una señal de entrada es `"00"` ó `"10"`, es par. Si una señal de entrada es `"01"` ó `"11"`, es impar. La salida de este circuito indica el número de señales pares existentes en la entrada (0, 1 ó 2). Por ejemplo, si las dos entradas son `"01"` y `"11"`, entonces la salida del circuito vale 0 (`"00"`); si las señales de entrada son `"00"` y `"01"`, entonces la salida del circuito vale 1 (`"01"`); si las dos entradas son `"00"` y `"10"`, entonces la salida del circuito vale 2 (`"10"`).

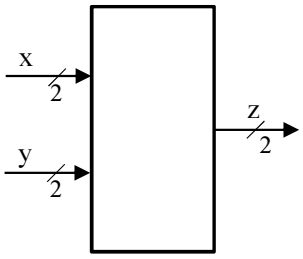

Figura 1.1: Entradas y salida del circuito del Ejercicio 1.

1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito digital empleando el mismo nombre para las señales que el mostrado en la Figura 1.1.

1.b) (1 punto) Escriba la tabla de la verdad del circuito digital. A partir de dicha tabla de la verdad, obtenga la función lógica que describe la salida (`z`) en función de las entradas (`x` e `y`). A continuación, escriba en VHDL una **architecture** que describa el *comportamiento* de un circuito que implemente dicha función lógica.

1.c) (0.5 puntos) Dibuje el diagrama al nivel de puertas lógicas de un circuito que implemente esta función. 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, la salida 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 entity contadorPar is
4   port ( z    : out std_logic_vector (1 downto 0);
5          x, y : in std_logic_vector (1 downto 0));
6 end entity contadorPar;
```

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

La tabla de verdad del circuito se muestra a continuación, donde – indica *don't care* y se ha empleado para hacer más compacta la tabla.

x(1)	x(0)	y(1)	y(0)	z(1)	z(0)
-	0	-	0	1	0
-	0	-	1	0	1
-	1	-	0	0	1
-	1	-	1	0	0

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

```
1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 architecture contadorPar of contadorPar is
4 begin
5    z(1) <= (not x(0)) and (not y(0));
6    z(0) <= x(0) xor y(0);
7 end architecture contadorPar;
```

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

La Figura 1.2 muestra el diagrama del circuito empleando una puerta AND, una puerta XOR y dos inversores.

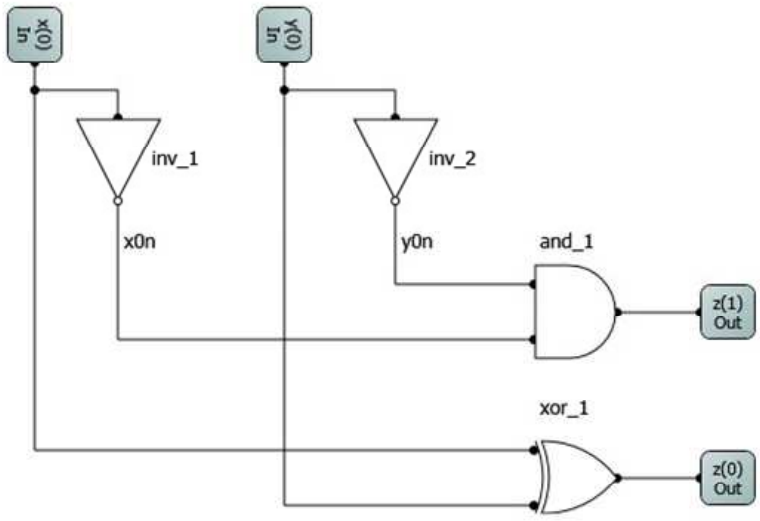

Figura 1.2: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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 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 xor2 is port
  ( y0     : out std_logic;
    x0, x1 : in  std_logic );
end entity xor2;

architecture xor2 of xor2 is
begin
  y0 <= x0 xor x1;
end architecture xor2;
```

Código VHDL 1.4: Apartado 1.c: puerta XOR 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.

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

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  architecture contadorPar_Estruc of contadorPar is
4    signal x0n, y0n: std_logic;
5  -- Declaración de las clases de los componentes
6    component and2 is
7      port (  y0     : out std_logic ;
8              x0, x1 : in std_logic);
9    end component and2;
10   component not1 is
11     port ( y0 : out std_logic;
12            x0 : in std_logic );
13   end component not1;
14   component xor2 is
15     port ( y0     : out std_logic;
16            x0, x1 : in std_logic );
17   end component xor2;
18 begin
19 -- Instanciación y conexión de los componentes
20   inv_1 : component not1 port map (x0n, x(0));
21   inv_2 : component not1 port map (y0n, y(0));
22   xor_1 : component xor2 port map (z(0), x(0), y(0));
23   and_1 : component and2 port map (z(1), x0n, y0n);
24 end architecture contadorPar_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, En las Figuras 1.3–1.4 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
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_contadorPar is
end entity bp_contadorPar;

architecture bp_contadorPar of bp_contadorPar is
  signal z : std_logic_vector (1 downto 0); --Conectar salidas UUT
  signal x, y: std_logic_vector(1 downto 0); --Conectar entradas UUT
  component contadorPar is port
   ( z : out std_logic_vector (1 downto 0);
     x, y : in std_logic_vector(1 downto 0));
  end component contadorPar;
begin
-- Instanciar y conectar UUT
  uut : component contadorPar port map
  ( z=> z, x => x, y => y);
  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
      y(1) <= test_in(3);
      y(0) <= test_in(2);
      x(1) <= test_in(1);
      x(0) <= test_in(0);
      wait for 10 ns;
      test_in := test_in + 1;
    end loop;
    wait;
  end process gen_vec_test;
end architecture bp_contadorPar;
```

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

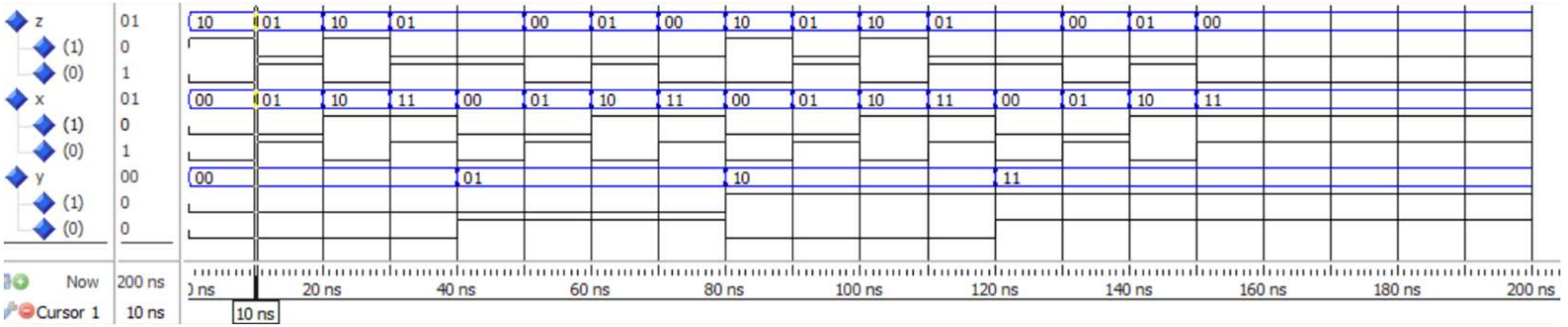

Figura 1.3: Apartado 1.e: simulación del banco de pruebas del diseño describiendo el comportamiento del circuito.

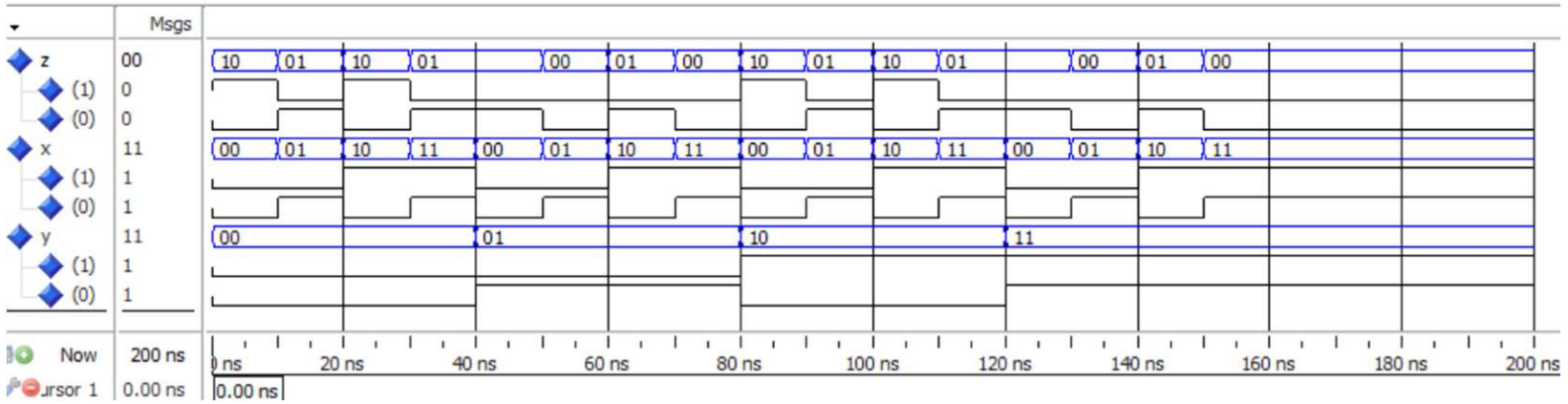

Figura 1.4: Apartado 1.e: simulación del banco de pruebas del diseño describiendo la estructura del circuito.

Ejercicio 2

En la Figura 1.5 se muestra el símbolo lógico de un circuito combinacional que tiene dos entradas `x1` y `x2` de ocho bits y una salida `z` de cuatro bits. El valor de la señal de salida `z` indica el número de posiciones en que las dos señales de entrada (`x1` y `x2`) tienen distinto valor. Por ejemplo, para las señales de entrada `"00010001"` y `"10010010"`, que difieren únicamente en los bits de tres posiciones, el circuito genera la señal de salida `"0011"` (valor 3 en decimal).

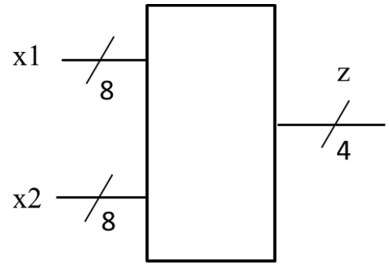

Figura 1.5: Entradas y salidas del circuito combinacional.

La **entity** del circuito se muestra a continuación.

```
entity circ is
  generic ( Width  : integer:=8;
            WidthZ : integer:=4);
  port ( z   : out std_logic_vector(WidthZ-1 downto 0);
         x1  : in  std_logic_vector(Width-1 downto 0);
         x2  : in  std_logic_vector(Width-1 downto 0) );
end entity circ;
```

2.a) (3 puntos) Diseñe en VHDL la **architecture** que describe el comportamiento del circuito combinacional descrito en el enunciado.

2.b) (3 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito. El banco de pruebas debe modificar el valor de las señales de entrada del circuito cada 10 ns. Además, 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. Emplee este banco de pruebas para comprobar el diseño realizado al contestar al Apartado 2.a.

Incluya en la memoria la captura de pantalla de los primeros 100 ns del cronograma obtenido al realizar la simulación del banco de pruebas con el simulador de VHDL.

Solución al Ejercicio 2

La **architecture** describiendo el comportamiento del circuito se muestra en Código VHDL 1.8.

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

architecture circ of circ is
signal aux: unsigned (Width-1 downto 0) ;
signal lev0_0, lev0_2, lev0_4 , lev0_6: unsigned (1 downto 0) ;
signal lev1_0, lev1_4 : unsigned(2 downto 0);
signal lev2: unsigned (3 downto 0) ;
begin
  aux    <= unsigned ( x1 xor x2);
  lev0_0 <= ( '0' & aux(0) ) + ( '0' & aux(1) );
  lev0_2 <= ( '0' & aux(2) ) + ( '0' & aux(3) );
  lev0_4 <= ( '0' & aux(4) ) + ( '0' & aux(5) );
  lev0_6 <= ( '0' & aux(6) ) + ( '0' & aux(7) );
  lev1_0 <= ( '0' & lev0_0 ) + ( '0' & lev0_2 );
  lev1_4 <= ( '0' & lev0_4 ) + ( '0' & lev0_6 );
  lev2   <= ( '0' & lev1_0 ) + ( '0' & lev1_4 );
  z      <= std_logic_vector( lev2 ) ;
end architecture circ;
```

Código VHDL 1.8: Apartado 2.a: **architecture** del circuito describiendo su comportamiento.

El banco de pruebas del circuito combinacional se muestra en el Código VHDL 1.9. El cronograma obtenidos al simular 100 ns del banco de pruebas, se muestra en la Figura 1.6.

```
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
  constant WIDTH : integer := 8;
  constant WIDTHZ : integer :=4;
  --Conectar salidas UUT
  signal z     : std_logic_vector(WIDTHZ-1 downto 0);
  --Conectar entradas UUT
  signal x1, x2  : std_logic_vector(WIDTH-1 downto 0);

  component circ is
    generic ( Width  : integer:= WIDTH;
              WidthZ : integer:= WIDTHZ );
    port (  z  : out std_logic_vector(WIDTHZ-1 downto 0);
           x1  : in  std_logic_vector(WIDTH-1 downto 0);
           x2  : in  std_logic_vector(WIDTH-1 downto 0) );
  end component circ;
```

```
23
24 begin
25   -- Instanciar y conectar UUT
26   uut : component circ
27        port map( z, x1, x2 );
28   gen_vec_test : process
29     -- Vectores test
30     variable test_in1     : unsigned (WIDTH-1 downto 0);
31     variable test_in2     : unsigned (WIDTH-1 downto 0);
32
33     variable sum          : unsigned (WIDTHZ-1 downto 0);
34     variable esperado_Z   : std_logic_vector(WIDTHZ-1 downto 0);
35     variable error_count  : integer   := 0;
36   begin
37     test_in1 := B"00000000";
38     report "Comienza la simulacion";
39     for count1 in 0 to 2**WIDTH-1 loop
40       test_in2 := B"00000000";
41       for count2 in 0 to 2**WIDTH-1 loop
42         x1 <= std_logic_vector(test_in1);
43         x2 <= std_logic_vector(test_in2);
44         sum := B"0000";
45         for i in 0 to WIDTH-1 loop
46            if (test_in1(i) /= test_in2(i)) then
47            sum := sum + "0001";
48            end if;
49         end loop;
50         esperado_Z := std_logic_vector(sum);
51         wait for 10 ns;
52         test_in2 := test_in2 + 1;
53         if ( esperado_Z /= z ) then
54           report "ERROR en la salida valida. Valor esperado: "&
55                  std_logic'image(esperado_Z(3))                &
56                  std_logic'image(esperado_Z(2))                &
57                  std_logic'image(esperado_Z(1))                &
58                  std_logic'image(esperado_Z(0))                &
59                  ", valor actual: "                            &
60                  std_logic'image(z(3))                         &
61                  std_logic'image(z(2))                         &
62                  std_logic'image(z(1))                         &
63                  std_logic'image(z(0))                         &
64                  " en el instante: "                           &
65                  time'image(now);
66           error_count := error_count + 1;
67         end if;
68       end loop;
69       test_in1 := test_in1 + 1;
70     end loop;
71     report  "ERROR: Hay "                &
72               integer'image(error_count) &
73               " errores.";
74     wait;
75   end process gen_vec_test;
76 end architecture bp_circ;
```

Código VHDL 1.9: Apartado 2.b: banco de pruebas del circuito.

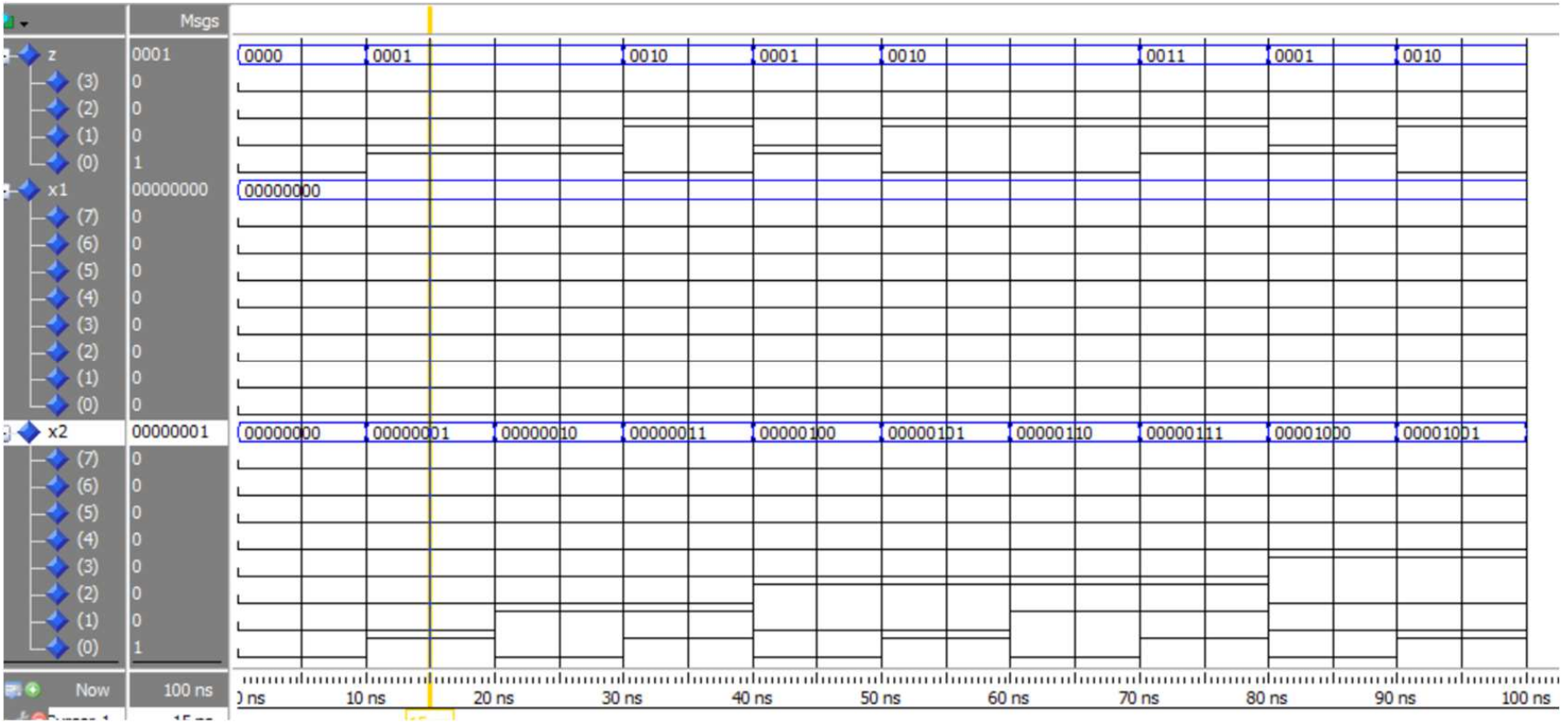

Figura 1.6: Cronograma del Apartado 2.b comprobando el comportamiento del circuito diseñado en el Apartado 2.a.