INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2018

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 `a`, `b` y `c`:

a	b	c	F1	F2
'0'	'0'	'0'	'0'	'0'
'0'	'0'	'1'	'0'	'0'
'0'	'1'	'0'	'0'	'1'
'0'	'1'	'1'	'0'	'1'
'1'	'0'	'0'	'1'	'0'
'1'	'0'	'1'	'1'	'0'
'1'	'1'	'0'	'0'	'1'
'1'	'1'	'1'	'1'	'0'

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 `a`, `b` y `c`, 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;
               a, b, c  : in std_logic );
end entity funcF1F2;
-------------------------------------
```

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

```
-------------------------------------
architecture Comp of funcF1F2 is
begin
 F1 <= a and (c or not b);
 F2 <= b and (not a or not c);
end architecture Comp;
-------------------------------------
```

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

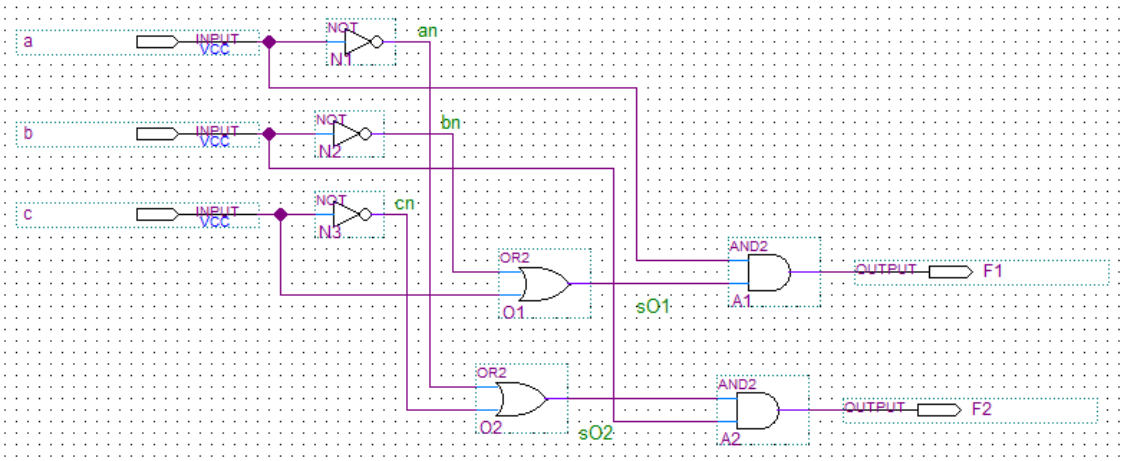

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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

El código VHDL de la **entity** y la **architecture** de estas cuatro puertas lógicas se muestra en Código VHDL 1.3, 1.4 y 1.5, respectivamente. El Código VHDL 1.6 muestra la **architecture** del circuito describiendo su estructura.

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

```
-------------------------------------
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 lógica.

```
  ------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcF1F2 is
   signal an, bn, cn: std_logic;
   signal sO1, sO2: 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 (an, a);
   N2 : component not1 port map (bn, b);
   N3 : component not1 port map (cn, c);
   O1 : component or2 port map (sO1, c, bn);
   O2 : component or2 port map (sO2, an, cn);
   A1 : component and2 port map (F1, sO1, a);
   A2 : component and2 port map (F2, sO2, b);
end architecture circuito_Estruc;
---------------------------------
```

Código VHDL 1.6: 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 1.7. 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 a, b, c   : std_logic;

    component funcF1F2 is
     port (   F1, F2 : out std_logic;
            a, b, c  : in std_logic );
     end component funcF1F2;

begin
    UUT : component funcF1F2 port map
          (F1, F2, a, b, c);

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);
          a <= std_logic(valor(2));
          b <= std_logic(valor(1));
          c <= std_logic(valor(0));
          wait for DELAY;
       end loop;
       wait;      -- Final de la simulación
   end process vec_test;
end architecture bp_funcF1F2;
-------------------------------------------------
```

Código VHDL 1.7: Solución al Apartado 1.e: banco de pruebas del circuito.

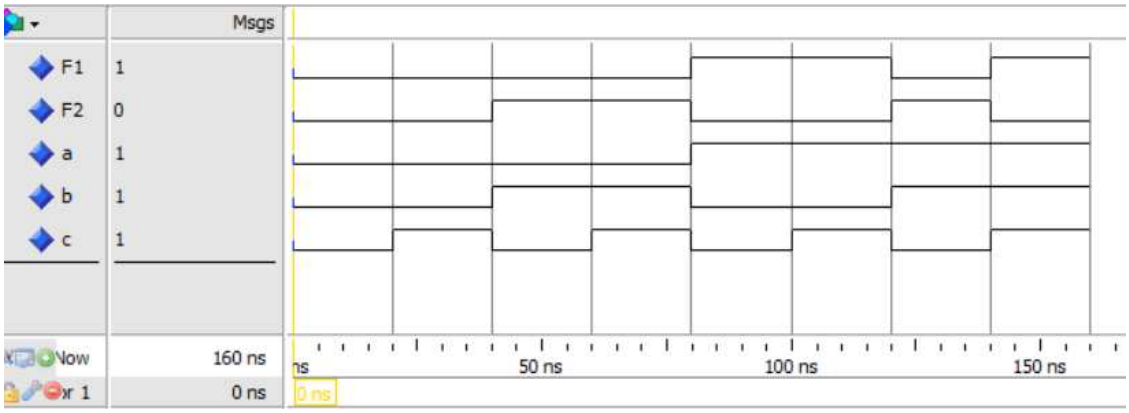

Figura 1.2: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.b.

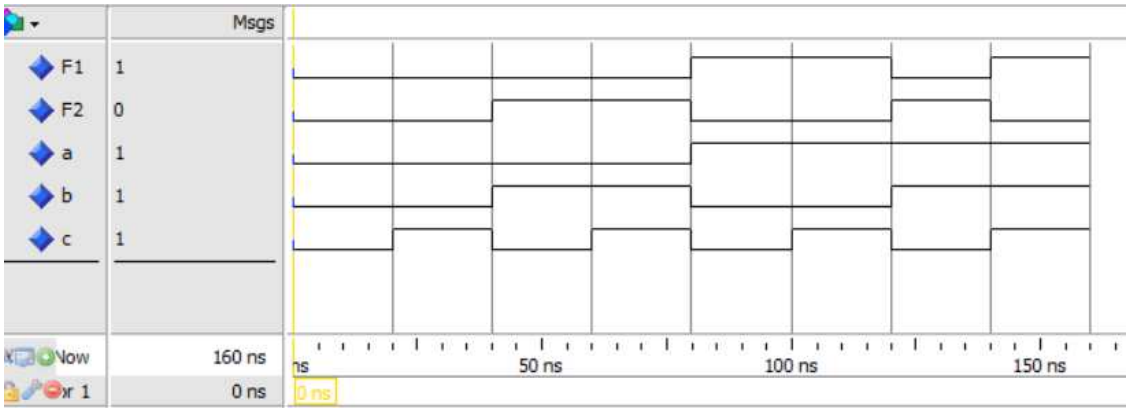

Figura 1.3: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.d.

EJERCICIO 2

Se quiere diseñar el circuito digital combinacional cuyo comportamiento se describe a continuación. El circuito tiene una señal de entrada `x` de 4 bits, una señal de entrada `shift` de un bit, dos señales de salida de un bit, `S1` y `S2`, y una señal de salida de 4 bits `y`.

La señal de salida `S1` vale '1' si la señal de entrada `x` interpretada como un número binario sin signo (N) tiene un valor comprendido entre 9 y 15 ($9 <= N <= 15$) y vale '0' en cualquier otro caso.

La señal de salida `S2` vale cero sólo si la señal de entrada `x` interpretada como un número binario sin signo tiene un valor cero o múltiplo de 2.

La señal de salida `y` es igual a la señal de entrada `x` si la señal de entrada `shift` vale 0. Mientras que si la señal de entrada `shift` vale '1', la señal de salida `y` se obtiene rotando la señal de entrada `x` una posición a la derecha.

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.

2.b) (1 punto) Dibuje el diagrama de un circuito que implemente las señales de salida `S1` y `S2` empleando puertas lógicas y además implemente la señal de salida `y` empleando cuatro multiplexores 2 a 1 de 1 bit. 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 todas las posibles entradas al circuito diseñado en los Apartados 2.a y 2.c. 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 dos cronogramas obtenidos al realizar la simulación del banco de pruebas de los circuitos diseñados en los Apartados 2.a y 2.c.

Solución al Ejercicio 2

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;
entity circ2E is
   port ( y: out std_logic_vector(3 downto 0);
          S1, S2: out std_logic;
          shift: in std_logic;
          x : in std_logic_vector(3 downto 0) );
end entity circ2E;

architecture Comp of circ2E is
begin
 process(x, shift)
 begin
    if (unsigned(x)<9) then
      S1 <= '0';
    else
      S1 <= '1';
    end if;
    S2 <= x(0);
    if (shift='0') then
      y <= x;
    else
      y <= x(0)&x(3)&x(2)&x(1);
    end if;
 end process;
end architecture Comp;
```

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

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 AND de dos entradas, OR de dos entradas y el multiplexor de 2 a 1 se muestra en Código VHDL 1.3, 1.4 y 1.9, respectivamente. El Código VHDL 1.10 muestra la **architecture** del circuito describiendo su estructura.

El banco de pruebas del circuito combinacional se muestra en Código VHDL 1.11–1.12. Los dos cronogramas obtenidos al simular el banco de pruebas usando como

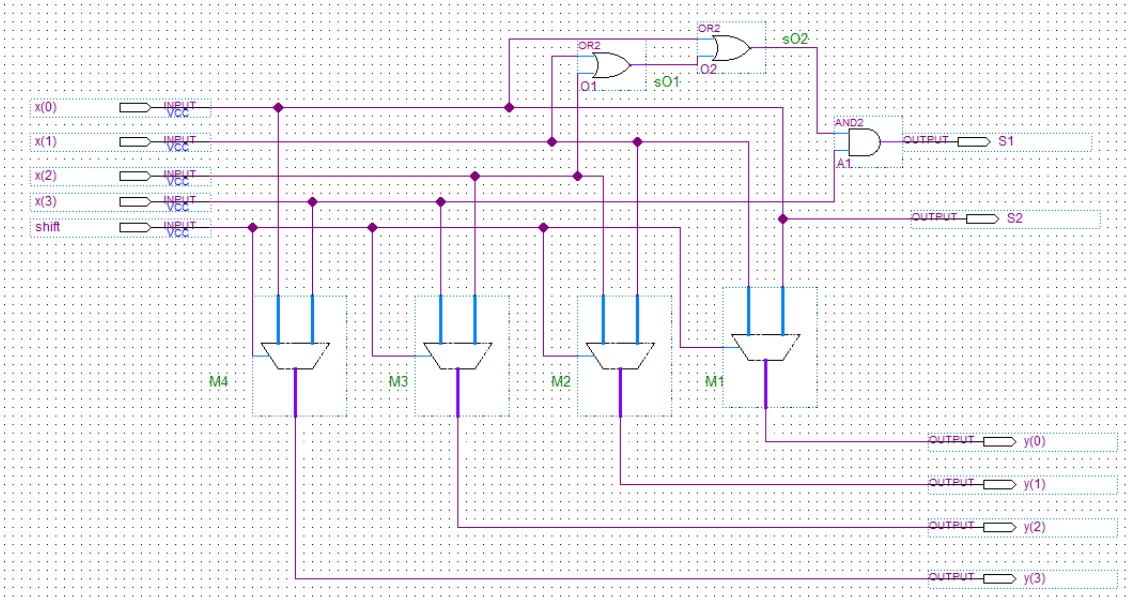

Figura 1.4: Solución al Apartado 2.b: diagrama al nivel de puertas lógicas.

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

entity mux2a1 is
   port ( y : out std_logic;
          x0, x1, s : in std_logic );
end entity mux2a1;

architecture mux2a1 of mux2a1 is
begin
   y <= x0 when (s='0')
     else x1;
end architecture mux2a1;
-------------------------------------
```

Código VHDL 1.9: Multiplexor 2 a 1.

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

```
 library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
-------------------------------------
architecture Comp of circ2E is
   signal sO1, sO2: 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 mux2a1 is
      port ( y : out std_logic;
                x0,x1,s : in std_logic );
   end component mux2a1;
   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 (sO1, x(2), x(1));
   O2 : component or2 port map (sO2, sO1, x(0));
   A1 : component and2 port map (S1, sO2, x(3));
   M4 : component mux2a1 port map (y(3), x(3), x(0),shift);
   M3 : component mux2a1 port map (y(2), x(2), x(3),shift);
   M2 : component mux2a1 port map (y(1), x(1), x(2),shift);
   M1 : component mux2a1 port map (y(0), x(0), x(1),shift);
   S2 <= x(0);
end architecture Comp;
-------------------------------------
```

Código VHDL 1.10: Solución al Apartado 2.c: descripción estructural del circuito.

```
  ------------------------------------------------------
-- Banco de pruebas del circuito del Ejercicio 2
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_circ2 is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_circ2;

architecture bp_circ2 of bp_circ2 is
    signal y, x : std_logic_vector(3 downto 0);
    signal shift, S1, S2   : std_logic;

    component circ2E is
    port ( y: out std_logic_vector(3 downto 0);
           S1, S2: out std_logic;
           shift: in std_logic;
           x : in std_logic_vector(3 downto 0) );
    end component circ2E;
  -- Procedure que calcula la salida (y_s, S1_s y S2_s) y lo compara con el
  -- valor de salida que se pasa como argumento (y, S1, S2)
  -- Si ambos valores no coinciden, se muestra un mensaje y se
  -- incrementa el contador de errores (error_count)
  procedure check_salida
      ( y      : in std_logic_vector(3 downto 0);
        S1, S2, shift          : in std_logic;
        x        : in std_logic_vector(3 downto 0);
        error_count: inout integer ) is
        variable y_esp: std_logic_vector(3 downto 0);
        variable S1_esp, S2_esp: std_logic;
  begin
      if (unsigned(x)<9) then S1_esp :='0';
      else S1_esp:='1';
      end if;
      if (x="0000" or to_integer(unsigned(x)) mod 2 =0 ) then S2_esp:='0';
      else S2_esp:='1';
      end if;
      if (shift='0') then y_esp:=x;
      else y_esp:= x(0)&x(3 downto 1);
      end if;

      assert( y = y_esp)
       report "ERROR en y. Entrada: " &   std_logic'image(x(3))
                & std_logic'image(x(2)) & std_logic'image(x(1)) &
                std_logic'image(x(0)) &
                ", resultado esperado: " & std_logic'image(y_esp(3))
                & std_logic'image(y_esp(2)) &
                std_logic'image(y_esp(1))    &    std_logic'image(y_esp(0))    &

                ", resultado actual: " & std_logic'image(y(3))
                &   std_logic'image(y(2)) &
                 std_logic'image(y(1)) & std_logic'image(y(0)) &
                " en el instante "
                & time'image(now);
```

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

```
        assert( S1 = S1_esp)
         report "ERROR en S1. Entrada: " &  std_logic'image(x(3))
                  & std_logic'image(x(2)) & std_logic'image(x(1)) &
                  std_logic'image(x(0)) &
                  ", resultado esperado: " &
                  std_logic'image(S1_esp)  &
                  ", resultado actual: " &
                   std_logic'image(S1) &
                  " en el instante " &
                  time'image(now);

        assert( S2 = S2_esp)
         report "ERROR en S2. Entrada: " &  std_logic'image(x(3))
                  & std_logic'image(x(2)) & std_logic'image(x(1)) &
                  std_logic'image(x(0)) &
                  ", resultado esperado: " &
                  std_logic'image(S2_esp)  &
                  ", resultado actual: " &
                   std_logic'image(S2) &
                  " en el instante " &
                  time'image(now);
      if (y /= y_esp)or (S1 /= S1_esp) or (S2 /= S2_esp) then
        error_count := error_count + 1;
      end if;

   end procedure check_salida;
   -- Fin de la definición del procedure


begin
    UUT : component circ2E port map
          (y, S1, S2, shift, x);

vec_test : process is
      variable error_count : integer := 0;
      variable temp: std_logic_vector(4 downto 0);
    begin
      report "Comienza la simulación";
      -- Generar todos los posibles valores de entrada
      for i in 0 to 2**5-1 loop
        temp := std_logic_vector(to_unsigned(i,5));
        x <= temp(3 downto 0);
        shift <= temp(4);
        wait for DELAY;
        check_salida(y, S1, S2, shift, x, error_count);
      end loop;

     report "Simulación finalizada con "
     & integer'image(error_count) &
     " errores";
      wait;      -- Final de la simulación
   end process vec_test;
end architecture bp_circ2;
-------------------------------------------------
```

Código VHDL 1.12: Continuación del banco de pruebas del circuito (solución del apartado 2.d).

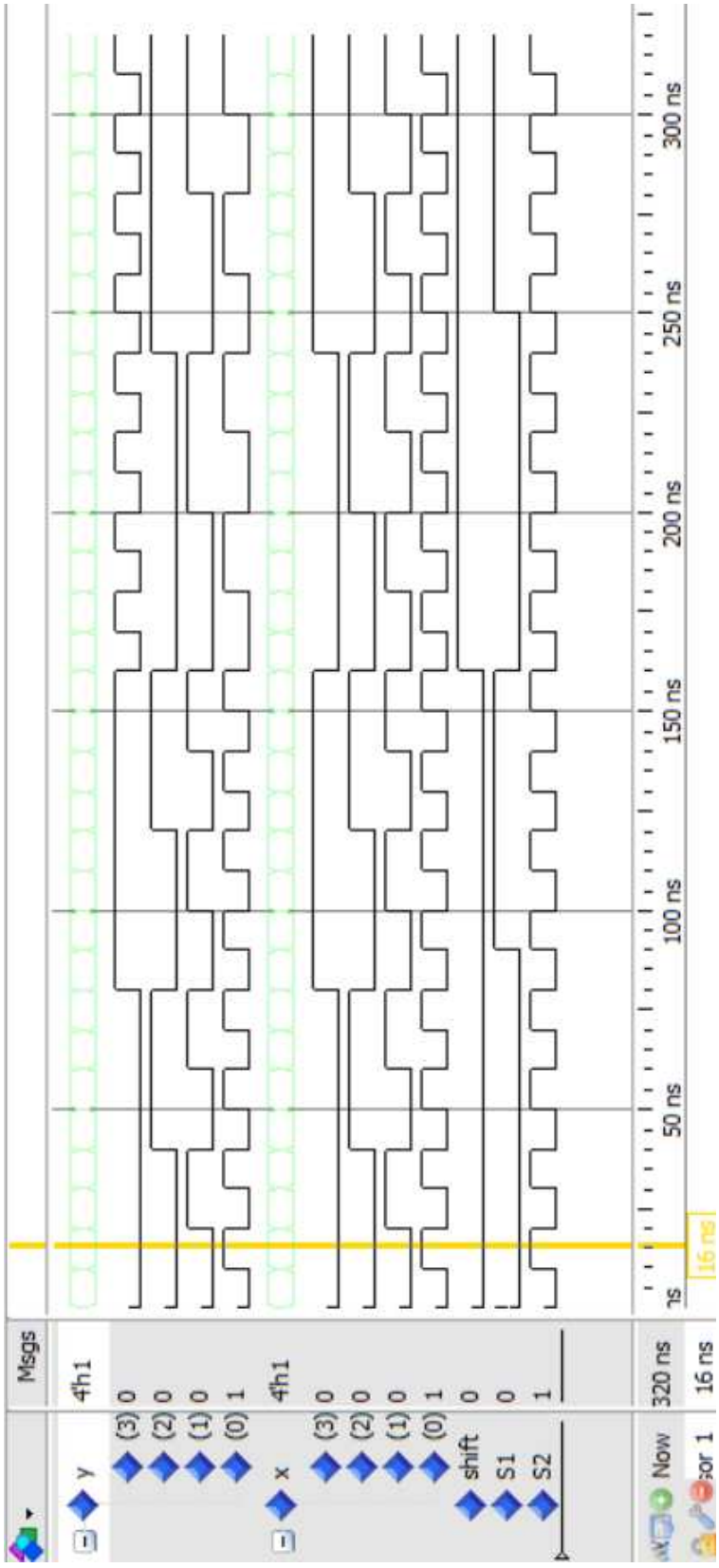

Figura 1.5: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.a.

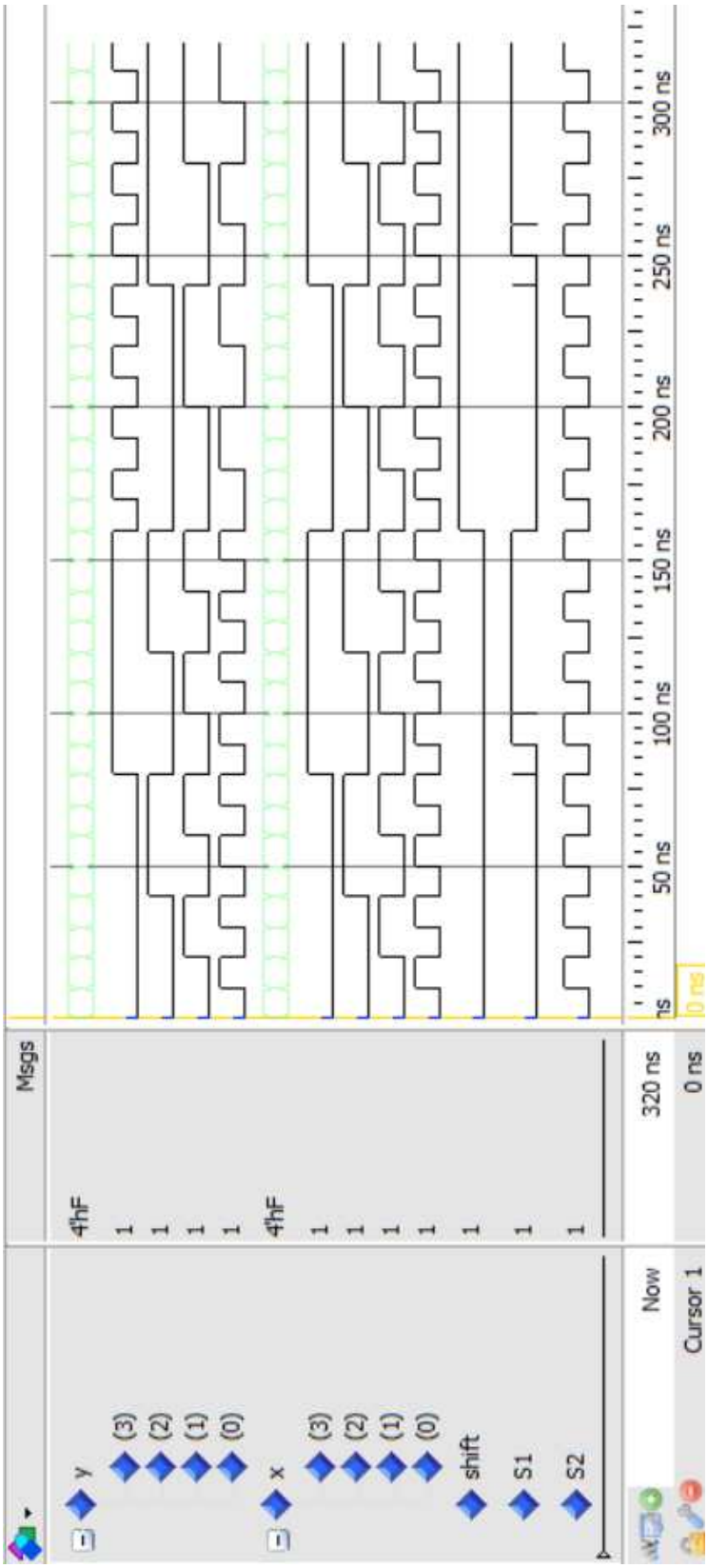

Figura 1.6: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.c.