INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2021

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'	'1'	'0'
'0'	'1'	'0'	'0'	'0'
'0'	'1'	'1'	'0'	'1'
'1'	'0'	'0'	'1'	'0'
'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 cuatro puertas AND de dos entradas, una puerta NOT, una puerta OR de dos entradas y una puerta OR de tres entradas.

El código VHDL de la **entity** y la **architecture** de estas cuatro puertas lógicas se muestra en Código VHDL 1.3, 1.4, 1.5 y 1.6, respectivamente. El Código VHDL 1.7 muestra la **architecture** del circuito describiendo su estructura.

```
-----------------------------------
--F1 = x+y'z
--F2 = xy+yz+xz

architecture Comp of funcF1F2 is
begin
  F1 <= x or (not y and z);
  F2 <= (x and y) or (y and z) or (x and z);
end architecture Comp;
-----------------------------------
```

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

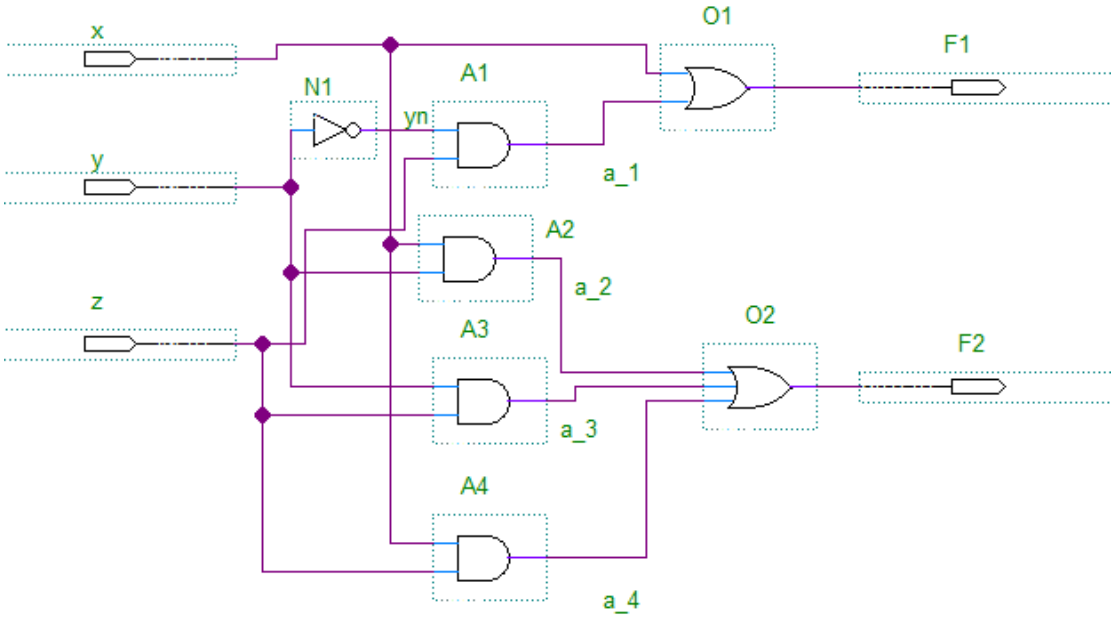

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

```
-----------------------------------
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 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 1.4: Puerta NOT 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 1.5: Puerta OR lógica de dos entradas.

```
-------------------------------------
-- OR de 3 entradas
library IEEE; use IEEE.std_logic_1164.all;

entity or3 is
  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 1.6: Puerta OR lógica de tres entradas.

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.8. Los dos cronogramas obtenidos al simular el banco de pruebas se muestran en las Figuras 1.2–1.3.

```
-----------------------------------
--F1 = x+y'z
--F2 = xy+yz+xz
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcF1F2 is
   signal yn, a_1, a_2, a_3, a_4: std_logic;
-- Declaración de las clases de los componentes
   component not1 is
      port (   y0 : out std_logic ;
               x0 : in std_logic);
   end component not1;

   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;

   component or3 is
      port ( y0 : out std_logic;
           x0, x1, x2 : in std_logic );
   end component or3;
begin
-- Instanciación y conexión de los componentes
   N1 : component not1 port map (yn, y);
   A1 : component and2 port map (a_1, yn, z);
   O1 : component or2 port map (F1, x, a_1);
   A2 : component and2 port map (a_2, x, y);
   A3 : component and2 port map (a_3, y, z);
   A4 : component and2 port map (a_4, x, z);
   O2 : component or3 port map (F2, a_2, a_3, a_4);
end architecture circuito_Estruc;
-----------------------------------
```

Código VHDL 1.7: Solución al Apartado 1.d: **architecture** del circuito describiendo su estructura.

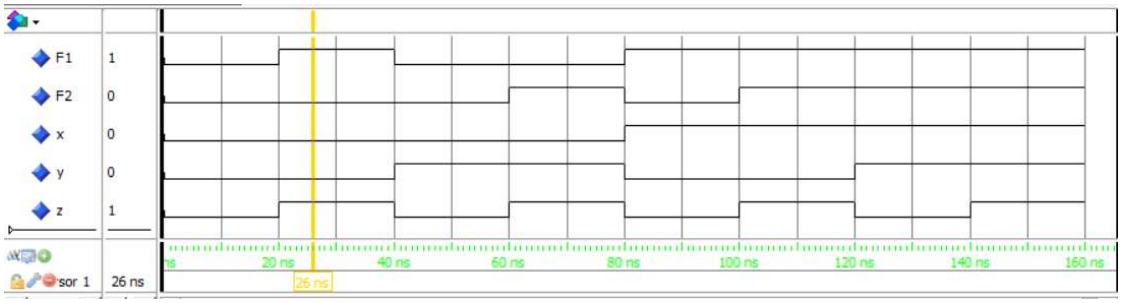

Figura 1.2: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.b.

```
-- 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 1.8: Solución al Apartado 1.e: banco de pruebas del circuito.

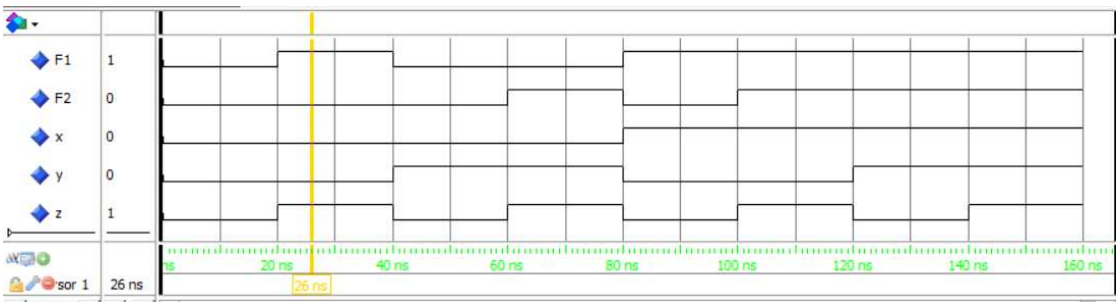

Figura 1.3: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.d.

EJERCICIO 2

Se quiere programar en VHDL un circuito combinacional que tiene dos señales de entrada de 8 bits llamadas `X` e `Y`, una señal de entrada de dos bits llamada `sel`, una señal de entrada de un bit llamada `E` y una señal de salida de 9 bits llamada `NUM1`.

El circuito ha de comportarse del modo siguiente. Si el valor de la señal `E` es '0', la señal de salida `NUM1` tiene valor cero. Si el valor de la señal `E` es '1', el valor de la señal de salida `NUM1` depende de los valores de las señales de entrada `X`, `Y` y `sel`.

La señal de entrada `sel` determina las operaciones que se realizan empleando los valores de las señales de entrada `X` e `Y`, cuyo resultado determina el valor de la señal de salida `NUM1`.

Si el valor de la señal `sel` es `"00"` y el valor de la señal `E` es '1', el valor de la señal de salida `NUM1` es igual al número de unos existentes en las señales de entrada `X` e `Y`. Por ejemplo, si la señal `X` tiene valor `"00000001"` y la señal `Y` tiene valor `"00000110"`, entonces la señal `NUM1` tiene como valor el número tres en su representación binaria con signo.

Si el valor de la señal `sel` es `"01"` y el valor de la señal `E` es '1', la señal de salida `NUM1` tiene como valor el número de las señales de entrada `X` e `Y` que son pares, considerando el valor cero como par. De este modo, si ambas señales son pares el valor de la señal `NUM1` es dos (`"000000010"`). Si ninguna de estas dos señales es par, entonces la señal `NUM1` tiene valor cero (`"000000000"`). Por último, si sólo una de estas dos señales es par, entonces la señal `NUM1` tiene valor uno (`"000000001"`).

Si el valor de la señal `sel` es `"10"` y el valor de la señal `E` es '1', la señal de salida `NUM1` tiene en el bit más significativo un valor '0' (`NUM1(8)<='0'`) y el valor del resto de bits son el resultado de la operación and lógica de las señales de entrada `X` e `Y` (`NUM1(7 downto 0) <= X and Y`).

Si el valor de la señal `sel` es `"11"` y el valor de la señal `E` es '1', la señal de salida `NUM1` tiene como valor el resultado de la suma de las señales de entrada `X` e `Y`, considerando que las señales `NUM1`, `X` e `Y` representan números binarios con signo.

2.a) (3 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. Emplee el convenio de especificar en primer lugar la señal de salida del circuito y posteriormente las señales de entrada. El número de bits de las señales `X`, `Y` y `NUM1` se ha de expresar como constante del tipo **generic**.

2.b) (3 puntos) Programe en VHDL un banco de pruebas que testee todas las posibles entradas al circuito diseñado en el Apartado 2.a.

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. El banco de pruebas debe mostrar al final del test un mensaje con el número total de errores detectados.

Incluya en la memoria los cronogramas obtenidos al realizar la simulación del banco de pruebas del circuito diseñado en el Apartado 2.a.

Solución al Ejercicio 2

La **entity** del circuito combinacional se muestra en Código VHDL 1.9.

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

entity circuito is
generic (n:integer:=8);
   port ( NUM1: out std_logic_vector ( n  downto 0);
               X, Y: in std_logic_vector (n-1  downto 0);
               sel  : in std_logic_vector(1  downto 0);
                    E: in std_logic );
end entity circuito;
-------------------------------------
```

Código VHDL 1.9: Solución al Apartado 2.a: **entity** del circuito.

El Código VHDL 1.10 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 Secuencial of circuito is
begin
   process(X, Y, E, sel)
      variable TEMP_CUENTA : INTEGER;
      variable xvar, yvar  : signed(n downto 0);
   begin
      if (E = '1') then
          TEMP_CUENTA :=  0;
          case sel is
              when "00" =>   --Numero de unos en X&Y
                  for I in n-1 downto 0 loop
                      if ( X(I) = '1') then
                          TEMP_CUENTA := TEMP_CUENTA + 1;
                      end if;
                      if ( Y(I) = '1') then
                          TEMP_CUENTA := TEMP_CUENTA + 1;
                      end if;
                  end loop;
                  NUM1 <= std_logic_vector(to_signed(TEMP_CUENTA,n+1));
              when "01" => --Numero de las señales x e y que son pares
                  if (X(0)= '0') then
                      TEMP_CUENTA := TEMP_CUENTA + 1;
                  end if;
                  if (Y(0)= '0') then
                      TEMP_CUENTA := TEMP_CUENTA + 1;
                  end if;
                  NUM1 <= std_logic_vector(to_signed(TEMP_CUENTA,n+1));
              when "10" => --X and Y
                  NUM1(n) <= '0';
                  NUM1(n-1 downto 0) <= X and Y;
              when others =>-- X+Y
                  xvar := resize(signed(X),n+1);
                  yvar := resize(signed(Y),n+1);
                  NUM1 <= std_logic_vector(xvar + yvar);
              end case;
      else
              NUM1 <=(others=>'0');
      end if;
   end process;
end architecture Secuencial;
-----------------------------------
```

Código VHDL 1.10: Solución al Apartado 2.a: circuito combinacional descrito usando un bloque **process**.

El banco de pruebas del circuito combinacional se muestra en Código VHDL 1.11–1.13. Los cronogramas obtenidos al simular el banco de pruebas usando como circuito a testear el diseño del Apartado 2.a se muestra en la Figura 1.4.

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

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

architecture bp_circuito of bp_circuito is
   constant WIDTH       :   integer := 8;
   signal NUM1       : std_logic_vector(WIDTH  downto 0);   -- Conectar salidas UUT
   signal E: std_logic;                                     -- Conectar entradas UUT
   signal sel: std_logic_vector(1  downto 0);
   signal X, Y   : std_logic_vector(WIDTH-1  downto 0);

component circuito is
generic (n:integer:=WIDTH);
   port ( NUM1: out std_logic_vector ( n  downto 0);
            X, Y: in std_logic_vector (n-1  downto 0);
            sel  : in std_logic_vector(1  downto 0);
                E: in std_logic );
end component circuito;

-- Procedure que calcula NUM1 (expected_NUM1) y lo compara con el
   -- valor de NUM1 que se pasa como argumento (actual_NUM1)
   -- Si ambos valores no coinciden, se muestra un mensaje y se
   -- incrementa el contador de errores (error_count)
   procedure check_circ
       ( i, j, k, l   : in integer;
         actual_NUM1 : in std_logic_vector (WIDTH       downto 0);
         error_count : inout integer ) is
      variable expected_NUM1 : integer;
      variable exp_NUM1: std_logic_vector(WIDTH  downto 0);
      variable xvar, yvar: std_logic_vector(WIDTH-1  downto 0);
      variable TEMP_CUENTA : INTEGER;
      variable iactual_NUM1 : integer;

   begin
```

Código VHDL 1.11: Solución al Apartado 2.b: banco de pruebas del circuito (1/3).

```
    if (l = 1) then
        expected_NUM1 :=  0;
     case k is
          when 0       =>
              xvar := std_logic_vector(TO_SIGNED(i,WIDTH));
              yvar := std_logic_vector(TO_SIGNED(j,WIDTH));
              for I in WIDTH-1 downto 0 loop
                  if ( xvar(I) = '1') then
                      expected_NUM1 := expected_NUM1 + 1;
                  end if;
                  if ( yvar(I) = '1') then
                      expected_NUM1 := expected_NUM1 + 1;
                  end if;
              end loop;
            expected_NUM1 := TO_INTEGER(TO_SIGNED(expected_NUM1,WIDTH+1));
          when 1       =>
                if ( i mod 2 = 0) then
                        expected_NUM1 := expected_NUM1 + 1;
                end if;
                if ( j mod 2 = 0) then
                        expected_NUM1 := expected_NUM1 + 1;
                end if;
                 expected_NUM1 := TO_INTEGER(TO_SIGNED(expected_NUM1,WIDTH+1));
          when 2       =>
                exp_NUM1(WIDTH-1 downto 0) :=   std_logic_vector(TO_SIGNED(i,WIDTH))
and
                                std_logic_vector(TO_SIGNED(j,WIDTH))     ;
                exp_NUM1(WIDTH) := '0';
                expected_NUM1 :=   TO_INTEGER(SIGNED(exp_NUM1));

                expected_NUM1 := TO_INTEGER(TO_SIGNED(expected_NUM1,WIDTH+1));
          when others =>
                expected_NUM1 := i+j;
                expected_NUM1 := TO_INTEGER(TO_SIGNED(expected_NUM1,WIDTH+1));
     end case;
     else
          exp_NUM1 := (others=>'0');
          expected_NUM1 :=   TO_INTEGER(SIGNED(exp_NUM1));
          expected_NUM1 := TO_INTEGER(TO_SIGNED(expected_NUM1,WIDTH+1));
     end if;

     iactual_NUM1 := TO_INTEGER(SIGNED(actual_NUM1));
          iactual_NUM1 := TO_INTEGER(TO_SIGNED(iactual_NUM1,WIDTH+1));
                -- Trunca el resultado a WIDTH+1 bits
```

Código VHDL 1.12: Solución al Apartado 2.b: continuación del banco de pruebas del circuito (2/3).

```
      assert( expected_NUM1 = iactual_NUM1 )
          report "ERROR. Operandos: " & integer'image(i) &
                    ", " & integer'image(j) &
                    ", Operacion: " & integer'image(k) &
                    ", resultado esperado: " &
                    integer'image(expected_NUM1) &
                    ", resultado actual: " &
                    integer'image(iactual_NUM1) &
                    " en el instante " &
                    time'image(now);
      if (expected_NUM1 /= iactual_NUM1) then
        error_count := error_count + 1;
      end if;
   end procedure check_circ;
   -- Fin de la definición del procedure
begin
   -- Instanciar y conectar UUT
   uut : component circuito
         port map( NUM1, X, Y, sel, E );
   main : process is
      variable error_count : integer := 0;
   begin

      report "Comienza la simulación";
      -- Vectores de test: operandos con valor próximo a cero
       E <='1';
       for l in 1 downto 0 loop
          for k in 0 to 3 loop
           for i in -2**(WIDTH-1) to 2**(WIDTH-1)-1 loop
              for j in -2**(WIDTH-1) to 2**(WIDTH-1)-1 loop
                 X   <= std_logic_vector(TO_SIGNED(i,WIDTH));
                 Y   <= std_logic_vector(TO_SIGNED(j,WIDTH));
                 sel <= std_logic_vector(TO_UNSIGNED(k,2));
                 wait for DELAY;
                 check_circ(i, j, k, l, NUM1, error_count);
              end loop;
           end loop;
         end loop;
         E<= '0';
         wait for DELAY;
      end loop;
     wait for DELAY;

      -- Informe mostrando el resultado del test
      if (error_count=0) then
          report "Finaliza la simulación: 0 errores";
      else
          report "Finaliza la simulación: " & integer'image(error_count) &
" errores";
      end if;

      wait;    -- Termina la simulación
   end process main;
end architecture bp_circuito;
```

Código VHDL 1.13: Solución al Apartado 2.b: continuación del banco de pruebas del circuito (3/3).

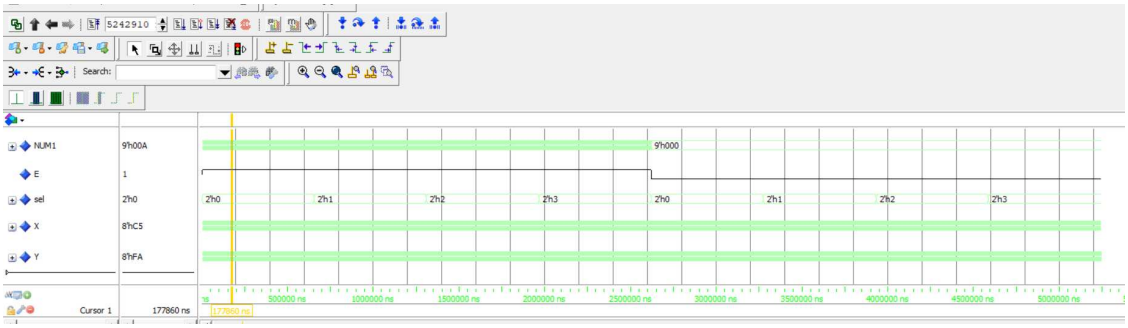

Figura 1.4: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el Apartado 2.a.