INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre 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'	'1'
'0'	'1'	'0'	'0'	'1'
'0'	'1'	'1'	'1'	'0'
'1'	'0'	'0'	'0'	'1'
'1'	'0'	'1'	'1'	'0'
'1'	'1'	'0'	'1'	'0'
'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 Apartados 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 tres puertas AND de dos entradas, dos puertas OR de dos entradas y y dos puertas XOR de dos entradas.

El código VHDL de la **entity** y la **architecture** de estas tres 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.

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

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

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

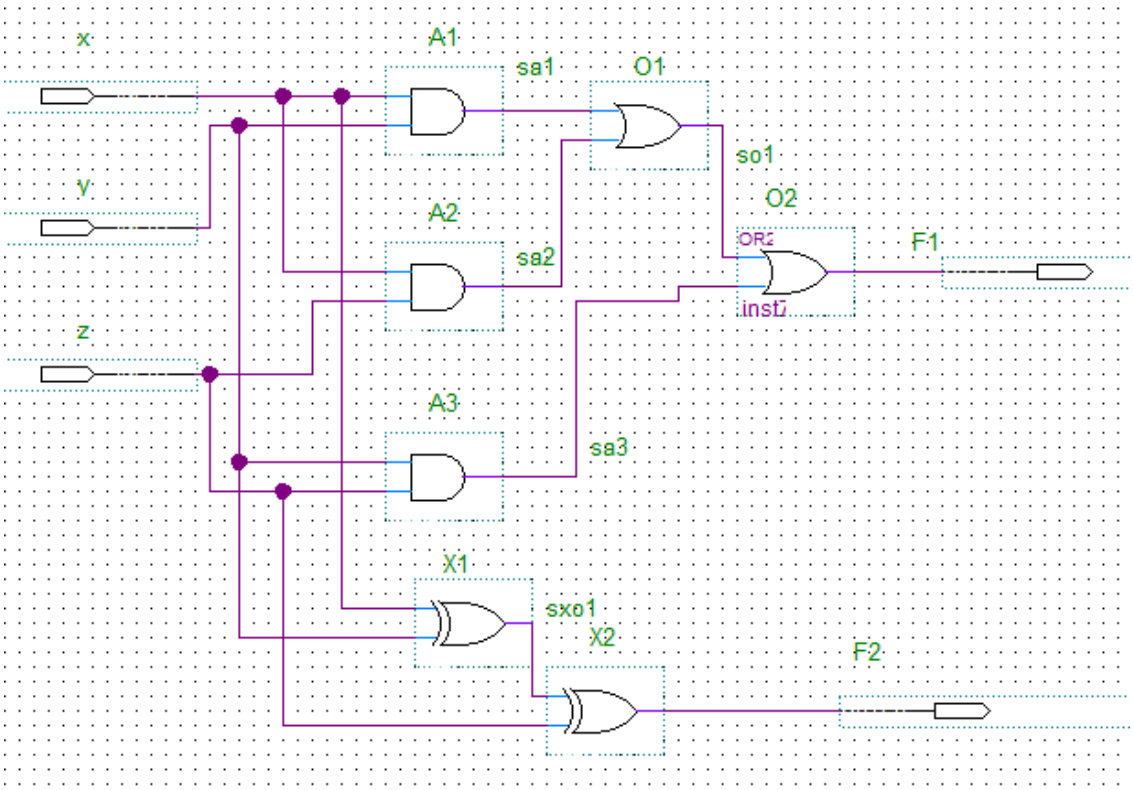

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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

```
-----------------------------------
-- XOR de 2 entradas
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 1.5: Puerta XOR lógica.

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.

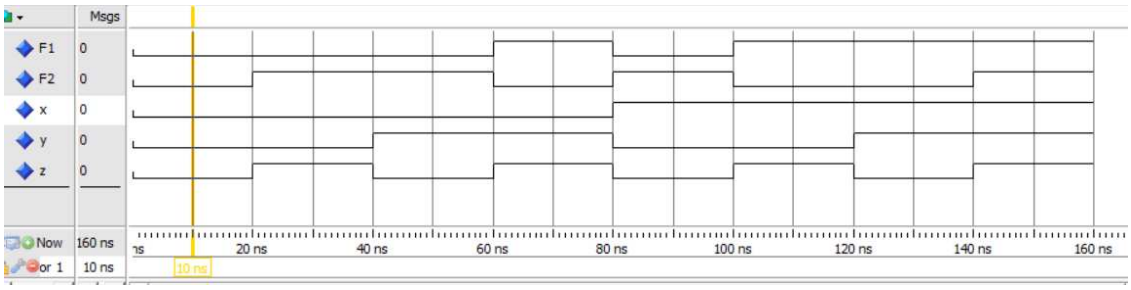

Figura 1.2: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.b.

```
-------------------------------------
--F1 = xy+xz+yz
--F2 = x xor y xor z
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcF1F2 is
   signal sa1, sa2, sa3: std_logic;
   signal so1, sxo1: 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;

   component xor2 is
      port ( y0 : out std_logic;
           x0, x1 : in std_logic );
   end component xor2;
begin
-- Instanciación y conexión de los componentes
   A1 : component and2 port map (sa1, x, y);
   A2 : component and2 port map (sa2, x, z);
   A3 : component and2 port map (sa3, y, z);
   O1 : component or2 port map (so1, sa1, sa2);
   O2 : component or2 port map (F1, so1, sa3);
   X1 : component xor2 port map (sxo1, x, y);
   X2 : component xor2 port map (F2, sxo1, z);

end architecture circuito_Estruc;
---------------------------------
```

Código VHDL 1.6: Solución al Apartado 1.d: **architecture** del circuito describiendo su estructura.

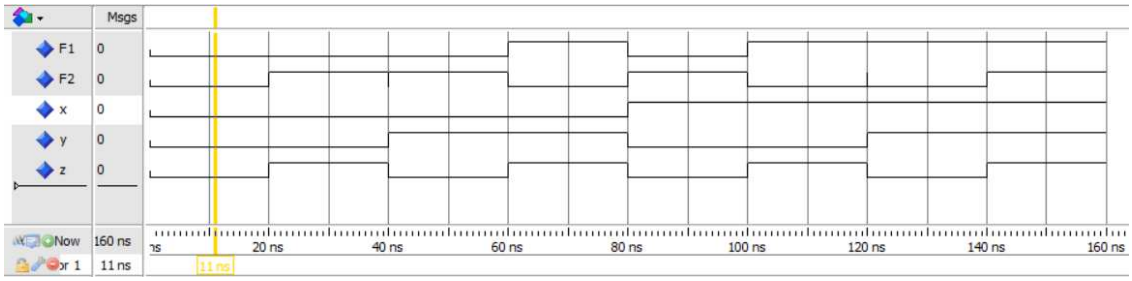

Figura 1.3: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.d.

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

EJERCICIO 2

Se quiere programar en VHDL una alarma de seguridad de un museo. La alarma está compuesta por dos circuitos, el circuito A y el circuito B. El circuito B controla el funcionamiento de 3 displays de 7 segmentos, que nos permiten visualizar si la alarma está habilitada o deshabilitada (mostrando las palabras on u off). En la Figura 1.4 se muestra el circuito de alarma, compuesto por los circuitos A y B y los 3 displays.

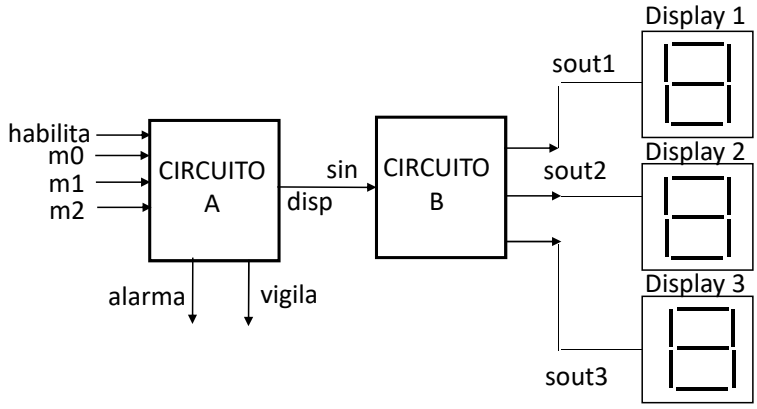

Figura 1.4: Alarma.

El circuito A tiene como entrada la señal de un bit `habilita` y 3 señales de entrada de un bit llamadas `m0`, `m1` y `m2`. Tiene como salida la señal de un bit `alarma`, la señal de dos bits `disp` y la señal de un bit `vigila`. La señal `disp` va a ser la entrada del circuito B.

El museo tiene 3 salas con un sensor de movimiento en cada una de ellas (señales `m0`, `m1` y `m2`). Mientras en la sala i ($i = 1, 2, 3$) se detecta movimiento, la señal `mi` tiene valor '1'. Por el contrario, si no se detecta movimiento en la sala i, la señal `mi` tiene valor '0'. En el museo hay un único guarda de seguridad.

Si se detecta movimiento en dos o más habitaciones y la alarma está habilitada, se pone a '1' la señal `alarma`. En caso contrario, la señal `alarma` está a '0'. Para habilitar/dehabilitar la alarma se da a la señal `habilita` el valor '1'/'0', respectivamente.

Si en el museo no se detecta movimiento en ninguna de las salas, se supone que el vigilante no está patrullando. Entonces, se pone la señal `vigila` a '1'. En cualquier otro caso, el valor de la señal `vigila` ha de ser '0'.

La señal `disp` toma los siguientes valores:

– La señal `disp` toma el valor "00" si hay que poner la palabra on en los diplays. Esto ocurre cuando la señal habilita está a '1'.

– La señal `disp` toma valor "01" si hay que poner la palabra off en los displays. Esto ocurre cuando la señal habilita está a '0',

– La señal `disp` toma los valores "10" o "11" si los tres displays se han de mantener apagados. Esto no sucede en el comportamiento normal de la alarma.

En la siguiente figura se muestra el display de 7 segmentos, donde se ha numerado cada segmento con números de 0 a 6. El circuito B se emplea para controlar el funcionamiento de los tres displays de siete segmentos. Este circuito tiene una señal de entrada de dos bits llamada `sin` y tres señales de salida de 7 bits llamadas `sout1`, `sout2` y `sout3`.

Para encender un determinado segmento del display i (i = 1,2,3) hay que poner a '1' el componente correspondiente de la señal `souti` (i = 1,2,3). Los componentes 6 a 0 de la señal `souti` se corresponden con los segmentos 6 a 0 del display i. Es decir, un valor de la señal `souti` igual a "0000001" indica que está iluminado el segmento 0 del display i y que el resto de los segmentos de dicho display están apagados.

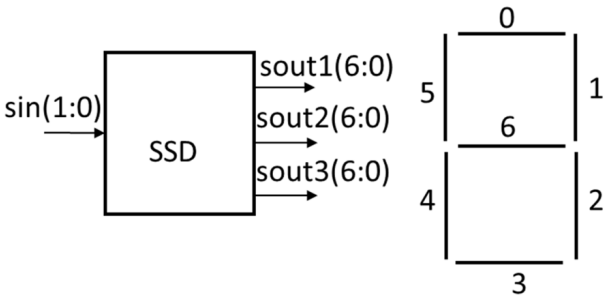

Figura 1.5: Circuito B y display de siete segmentos.

Los valores que han de tener las señales de salida `sout1`, `sout2` y `sout3` dependen de los valores de la señal de entrada `sin`:

– Si la señal de entrada `sin` tiene el valor "00", el display 1 tiene que estar apagado, en el display 2 se ha de mostrar la o y en el display 3 se ha de mostrar la n.

- Si la señal de entrada `sin` tiene el valor `"01"`, el display 1 tiene que mostrar la o, el display 2 tiene que mostrar la f y el display 3 tiene que mostrar la f.
- Para cualquier otro valor de la señal `sin`, los tres displays tienen que estar apagados.

2.a) (2 puntos) Obtenga para el circuito A las funciones lógicas de las señales de salida de dicho circuito en función de sus señales de entrada. Dibuje el diagrama de un circuito que implemente las funciones lógicas obtenidas 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 la **entity** y la **architecture** que describe el circuito que ha dibujado anteriormente. 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) (1.5 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el funcionamiento del circuito B empleando un bloque **process** y sentencias **case**. 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.c) (0.5 puntos) Escriba en VHDL la **entity** y **architecture** que describa el circuito completo. El circuito completo se tiene que describir instanciando y conectando adecuadamente los circuitos A y B diseñados en los apartados 2.a y 2.b, respectivamente. 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.d) (2 puntos) Programe en VHDL un banco de pruebas que testee el circuito diseñado en el Apartado 2.c para todos los posibles valores de las entradas. 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 el cronograma obtenido al realizar la simulación del banco de pruebas del circuito diseñado en el Apartados 2.c.

Solución al Ejercicio 2

La Figura 1.6 muestra el diagrama del circuito implementado empleando cuatro puertas AND de dos entradas, una puerta AND de tres entradas, cuatro inversores y una puertas OR de tres entradas.

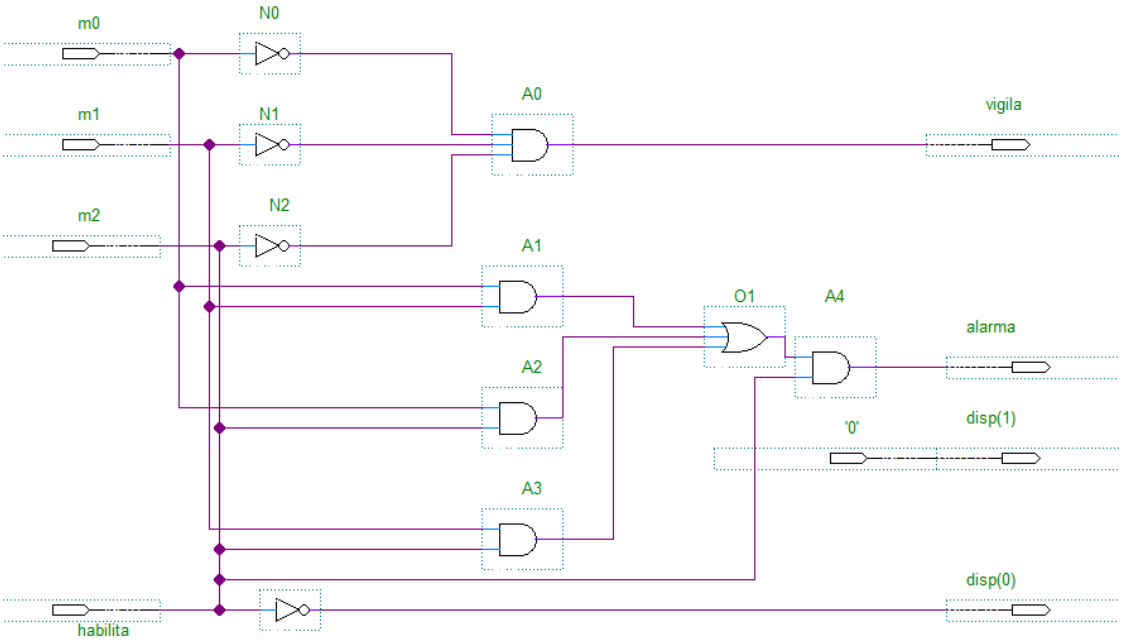

Figura 1.6: Solución al Apartado 2.a: diagrama al nivel de puertas lógicas.

El código VHDL de la **entity** y la **architecture** de estas cuatro puertas lógicas se muestra en Código VHDL 1.3, 1.8, 1.9 y 1.10, respectivamente.

```
----------------------------------
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 1.8: Puerta AND lógica de tres entradas.

```
-------------------------------------
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 1.9: Puerta inversora.

```
-------------------------------------
-- 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 1.10: Puerta OR lógica de tres entradas.

El Código VHDL 1.11 muestra el diseño del circuito describiendo su estructura.

El Código VHDL 1.12 muestra el código VHDL del circuito B empleando sólo un bloque **process** y sentencias secuenciales.

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

entity circuitoA is
   port ( alarma, vigila: out std_logic;
            disp: out std_logic_vector(1 downto 0);
            habilita: in std_logic;
            m0, m1, m2: in std_logic );
end entity circuitoA;
architecture Comp of circuitoA is
   signal nm0, nm1, nm2: std_logic;
   signal sa1, sa2, sa3,so1: std_logic;
-- Declaración de las clases de los componentes
   component and3 is
      port (   y0 : out std_logic ;
            x0, x1, x2: in std_logic);
   end component and3;

   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;

component not1 is
      port ( y0 : out std_logic;
                x0 : in std_logic );
   end component not1;
begin
   N0 : component not1 port map (nm0, m0);
   N1 : component not1 port map (nm1, m1);
   N2 : component not1 port map (nm2, m2);
   A0 : component and3 port map (vigila, nm0, nm1, nm2);
   A1 : component and2 port map (sa1, m0, m1);
   A2 : component and2 port map (sa2, m0, m2);
   A3 : component and2 port map (sa3, m1, m2);
   O1 : component or3 port map (so1, sa1, sa2, sa3);
   A4 : component and2 port map (alarma, habilita, so1);
   N3 : component not1 port map (disp(0), habilita);
   disp(1) <= '0';

end architecture Comp;
-------------------------------------
```

Código VHDL 1.11: Diseño del circuito A describiendo su estructura.

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

entity circuitoB is
   port ( sout1, sout2, sout3: out std_logic_vector(6 downto 0);
             sin: in std_logic_vector(1 downto 0) );
end entity circuitoB;
architecture Comp of circuitoB is
begin
   process(sin)
   begin
      case sin is
          when "00" => -- on
              sout1 <= "0000000";
              sout2 <= "0111111";--o
              sout3 <= "0110111";--n
          when "01" => -- off
              sout1 <= "0111111";--o
              sout2 <= "1110001";--f
              sout3 <= "1110001";--f
          when others => -- todos apagados
              sout1 <= "0000000";
              sout2 <= "0000000";
              sout3 <= "0000000";
     end case;
   end process;

end architecture Comp;
-------------------------------------
```

Código VHDL 1.12: Solución al Apartado 2.b: circuito B descrito usando un bloque **process**.

El Código VHDL 1.13 muestra el código VHDL del circuito completo descrito mediante la conexión del circuito A y B.

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

entity circuitoC is
   port ( alarma, vigila: out std_logic;
            sout1, sout2, sout3: out std_logic_vector(6 downto 0);
            habilita: in std_logic;
            m0, m1, m2: in std_logic );
end entity circuitoC;
architecture Comp of circuitoC is
   signal sdisp: std_logic_vector(1 downto 0);
-- Declaración de las clases de los componentes
   component circuitoA is
      port ( alarma, vigila: out std_logic;
            disp: out std_logic_vector(1 downto 0);
            habilita: in std_logic;
            m0, m1, m2: in std_logic);
   end component circuitoA;

   component circuitoB is
      port ( sout1, sout2, sout3: out std_logic_vector(6 downto 0);
            sin: in std_logic_vector(1 downto 0));
   end component circuitoB;

begin
   A1 : component circuitoA port map(alarma, vigila, sdisp, habilita, m0,
m1, m2);
   B1 : component circuitoB port map (sout1, sout2, sout3, sdisp);

end architecture Comp;
-------------------------------------
```

Código VHDL 1.13: Solución al Apartado 2.c: circuito C descrito mediante conexión de los circuitos A y B.

El banco de pruebas del circuito combinacional se muestra en Código VHDL 1.14–1.17. El cronograma obtenido al simular el banco de pruebas usando como circuito a testear el diseño del apartado 2.c se muestra en la Figura 1.7.

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

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

architecture bp_circuitoC of bp_circuitoC is
    signal alarma, vigila: std_logic;
    signal sout1, sout2, sout3: std_logic_vector(6 downto 0);
    signal habilita, m0, m1, m2: std_logic;

    component circuitoC is
     port (    alarma, vigila: out std_logic;
           sout1, sout2, sout3: out std_logic_vector(6 downto 0);
           habilita: in std_logic;
           m0, m1, m2: in std_logic );
     end component circuitoC;
 procedure check_salida
      ( habilita     : in std_logic;
        m0, m1, m2   : in std_logic;
        sout1, sout2, sout3: in std_logic_vector(6 downto 0);
        alarma, vigila: in std_logic;
        error_count: inout integer ) is
     variable alarma_ESP, vigila_ESP: std_logic;
     variable sout1_ESP, sout2_ESP, sout3_ESP: std_logic_vector(6 downto 0);
    begin
      sout1_ESP := "0000000";
      sout2_ESP := "0000000";
      sout3_ESP := "0000000";
      vigila_ESP := not(m0 or m1 or m2);
      alarma_ESP := '0';

      if (((m0 and m1)or(m0 and m2)or (m1 and m2)or(m0 and m1 and m2))='1')
then
        if (habilita ='1') then
          alarma_ESP := '1';
        else
          alarma_ESP := '0';
        end if;
      end if;
      if (habilita='1' ) then
        sout1_ESP := "0000000";
        sout2_ESP := "0111111";
        sout3_ESP := "0110111";
      elsif (habilita ='0') then
        sout1_ESP := "0111111";
        sout2_ESP := "1110001";
        sout3_ESP := "1110001";
       end if;
```

Código VHDL 1.14: Solución al Apartado 2.d: banco de pruebas del circuito (1/4).

```
        assert(  alarma = alarma_ESP)
        report "ERROR. Entrada: " & std_logic'image(habilita)
                  & std_logic'image(m0) & std_logic'image(m1) &
                  std_logic'image(m2) &
                  ",resultado esperado de alarma: " &
                   std_logic'image(alarma_ESP) &
                   ",resultado actual: "
                  &   std_logic'image(alarma) &
                  " en el instante "
                  & time'image(now);

        assert(  vigila = vigila_ESP)
        report "ERROR. Entrada: " & std_logic'image(habilita)
                  & std_logic'image(m0) & std_logic'image(m1) &
                  std_logic'image(m2) &
                  ",resultado esperado de vigila: " &
                   std_logic'image(vigila_ESP) &
                  ",resultado actual: "
                  &   std_logic'image(vigila) &
                  " en el instante "
                  & time'image(now);

        assert(  sout1 = sout1_ESP)
        report "ERROR. Entrada: " & std_logic'image(habilita)
                  & std_logic'image(m0) & std_logic'image(m1) &
                  std_logic'image(m2) &
                  ",resultado esperado de sout1: " &
                   std_logic'image(sout1_ESP(6)) &
                  std_logic'image(sout1_ESP(5)) & std_logic'image(sout1_ESP(4))
&
                  std_logic'image(sout1_ESP(3)) & std_logic'image(sout1_ESP(2))
&
                  std_logic'image(sout1_ESP(1)) & std_logic'image(sout1_ESP(0))
&

                  ",resultado actual: " &
                  std_logic'image(sout1(6)) &
                  std_logic'image(sout1(5)) & std_logic'image(sout1(4)) &
                  std_logic'image(sout1(3))    &    std_logic'image(sout1(2))    &

                  std_logic'image(sout1(1))    &    std_logic'image(sout1(0))    &

                  " en el instante "
                  & time'image(now);
```

Código VHDL 1.15: Solución al Apartado 2.d: banco de pruebas del circuito (2/4).

```
        assert( sout2 = sout2_ESP)
        report "ERROR. Entrada: " & std_logic'image(habilita)
                & std_logic'image(m0) & std_logic'image(m1) &
                std_logic'image(m2) &
                ", resultado esperado de sout2: " &
                 std_logic'image(sout2_ESP(6)) &
                std_logic'image(sout2_ESP(5)) & std_logic'image(sout2_ESP(4))
&
                std_logic'image(sout2_ESP(3)) & std_logic'image(sout2_ESP(2))
&
                std_logic'image(sout2_ESP(1)) & std_logic'image(sout2_ESP(0))
&

                ", resultado actual: " &
                std_logic'image(sout2(6)) &
                std_logic'image(sout2(5)) & std_logic'image(sout2(4)) &
                std_logic'image(sout2(3))   &   std_logic'image(sout2(2))   &

                std_logic'image(sout2(1))   &   std_logic'image(sout2(0))   &

                " en el instante "
                & time'image(now);

        assert( sout3 = sout3_ESP)
        report "ERROR. Entrada: " & std_logic'image(habilita)
                & std_logic'image(m0) & std_logic'image(m1) &
                std_logic'image(m2) &
                ", resultado esperado de sout3: " &
                 std_logic'image(sout3_ESP(6)) &
                std_logic'image(sout3_ESP(5)) & std_logic'image(sout3_ESP(4))
&
                std_logic'image(sout3_ESP(3)) & std_logic'image(sout3_ESP(2))
&
                std_logic'image(sout3_ESP(1)) & std_logic'image(sout3_ESP(0))
&

                ", resultado actual: " &
                std_logic'image(sout3(6)) &
                std_logic'image(sout3(5)) & std_logic'image(sout3(4)) &
                std_logic'image(sout3(3))   &   std_logic'image(sout3(2))   &

                std_logic'image(sout3(1))   &   std_logic'image(sout3(0))   &

                " en el instante "
                & time'image(now);
```

Código VHDL 1.16: Solución al Apartado 2.d: banco de pruebas del circuito (3/4).

```
      if (alarma/= alarma_ESP) or (vigila /= vigila_ESP)
          or (sout1 /= sout1_ESP) or (sout2/=sout2_ESP) or (sout3/=sout3_ESP)
then
        error_count := error_count + 1;
      end if;

   end procedure check_salida;
   -- Fin de la definición del procedure
begin
    UUT : component circuitoC  port map
          (alarma, vigila, sout1, sout2, sout3, habilita, m0, m1, m2);
gen_vec_test : process is
     variable valor   : unsigned (3  downto 0);
     variable error_count          : integer     := 0;
    begin
       -- Generar todos los posibles valores de entrada
       for i  in 0  to 2**4-1  loop
          valor := to_unsigned(i,4);
          habilita <= std_logic(valor(3));
          m0 <= std_logic(valor(2));
          m1 <= std_logic(valor(1));
          m2 <= std_logic(valor(0));
          wait for DELAY;
          check_salida(habilita, m0, m1, m2, sout1, sout2, sout3, alarma,
vigila, error_count);
       end loop;
       report "Simulación finalizada con "
      & integer'image(error_count) &
      " errores";
       wait;      -- Final de la simulación     end process vec_test;
   end process gen_vec_test;
end architecture bp_circuitoC;
-------------------------------------------------
```

Código VHDL 1.17: Solución al Apartado 2.d: banco de pruebas del circuito (4/4).

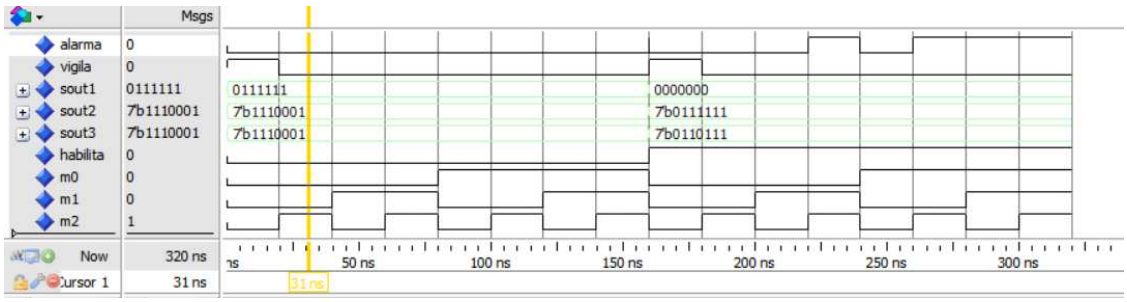

Figura 1.7: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.c.