INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Junio de 2019

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'	'1'
'0'	'1'	'0'	'1'	'0'
'0'	'1'	'1'	'1'	'1'
'1'	'0'	'0'	'1'	'0'
'1'	'0'	'1'	'1'	'1'
'1'	'1'	'0'	'1'	'0'
'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 or b;
 F2 <= c and (not a or not b);
end architecture Comp;
-------------------------------------
```

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

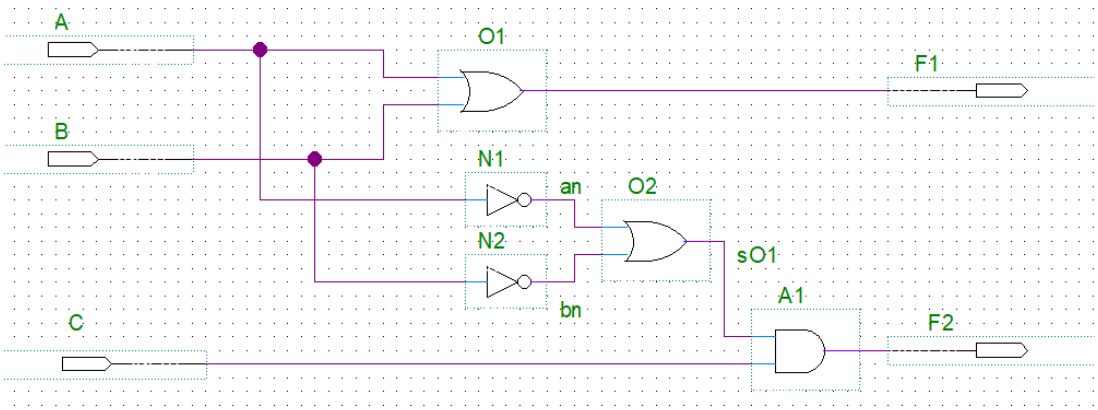

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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

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.

```
-------------------------------------
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: std_logic;
   signal sO1: 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);
   O1 : component or2 port map (F1, a, b);
   O2 : component or2 port map (sO1, an, bn);
   A1 : component and2 port map (F2, c, sO1);
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.

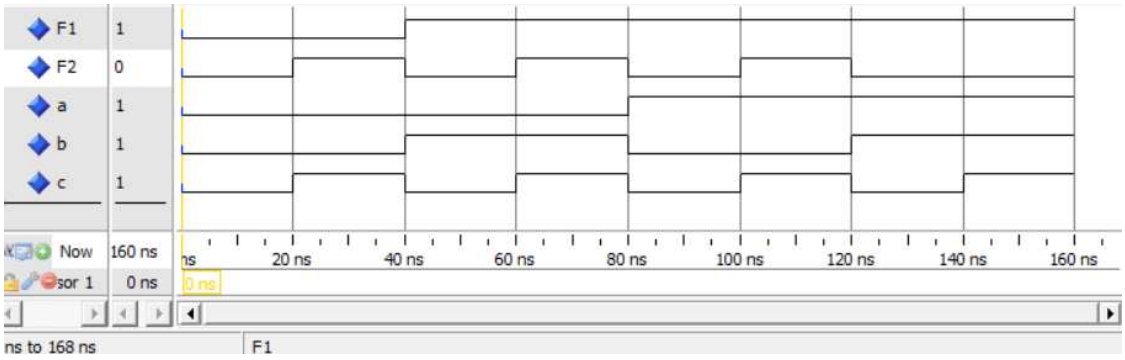

Figura 1.2: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.b.

```
  -- 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 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.

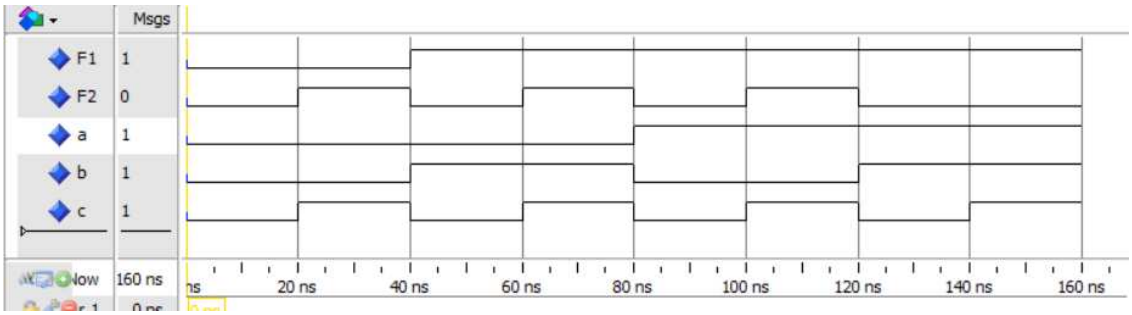

Figura 1.3: Cronograma del Apartado 1.e para el circuito de test del Apartado 1.d.

EJERCICIO 2

Se quiere diseñar el circuito digital combinacional cuyo comportamiento se describe a continuación. El circuito tiene una señal de entrada `mes` de 4 bits, una señal de entrada `bisiesto` de un bit, y cuatro señales de salida de un bit, `dias28`, `dias29`, `dias30` y `dias31`.

La señal de entrada `mes` indica el número binario correspondiente a un mes del año. Es decir, cuando dicha sale vale "0001" se corresponde al mes de enero, "0010" a febrero,..., y "1100" a diciembre. La señal de entrada `bisiesto` tiene valor '1' si y sólo si ese año es bisiesto.

La señal de salida `dias28` vale '1' si la señal de entrada `mes` interpretada como un número binario sin signo se corresponde con el mes de febrero y además ese año no es bisiesto. La señal de salida `dias28` vale '0' en cualquier otro mes del año o si es febrero de un año bisiesto.

La señal de salida `dias29` vale '1' si el año es bisiesto y la señal de entrada `mes` se corresponde con febrero. La señal de salida `dias29` vale '0' en cualquier otro mes del año o si es febrero de un año no bisiesto.

La señal de salida `dias30` vale '1' si la señal de entrada `mes` interpretada como un número binario sin signo se corresponde con un mes que tenga 30 días y vale '0' el resto de meses del año.

La señal de salida `dias31` vale '1' si la señal de entrada `mes` interpretada como un número binario sin signo se corresponde con un mes que tenga 31 días y vale '0' el resto de meses del año.

No importa el valor que toman las señales de salida del circuito cuando la señal de entrada `mes` no se corresponde con un mes del año. Es decir, cuando la señal `mes` tiene un valor que no está comprendido entre "0001" y "1100".

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) Escriba las tablas de verdad correspondientes a las señales de salida del circuito. Dibuje el diagrama de un circuito que implemente las señales de salida empleando 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.

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

La **entity** del circuito combinacional se muestra en Código VHDL 1.8.

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

entity circuito is
   port ( dias28, dias29, dias30, dias31: out std_logic;
          mes : in std_logic_vector(3 downto 0 );
          bisiesto: in std_logic);
end entity circuito;
-------------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.a: **entity** del circuito.

El Código VHDL 1.9 muestra el código VHDL del circuito combinacional empleando sólo un bloque **process** y sentencias secuenciales.

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, inversor y la puerta AND de tres entradas se muestra en Código VHDL 1.3, 1.4, 1.5 y 1.10, respectivamente. El Código VHDL 1.11 muestra la **architecture** del circuito describiendo su estructura.

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

architecture circuitoIF of circuito is
begin
    process ( mes, bisiesto )
    begin
        dias28 <= '0';
        dias29 <= '0';
        dias30 <= '0';
        dias31 <= '0';
        case mes is
           when "0001" =>   --Enero
                dias31 <= '1';
           when "0010" =>   --Febrero
                if (bisiesto = '1') then
                 dias29 <= '1';
                else
                 dias28 <= '1';
                end if;
            when "0011" =>   --Marzo
                 dias31 <= '1';
            when "0100" =>   --Abril
                 dias30 <= '1';
            when "0101" =>   --Mayo
                 dias31 <= '1';
            when "0110" =>   --Junio
                 dias30 <= '1';
            when "0111" =>   --Julio
                 dias31 <= '1';
            when "1000" =>   --Agosto
                 dias31 <='1';
            when "1001" =>   --Septiembre
                 dias30 <= '1';
            when "1010" =>   --Octubre
                  dias31 <= '1';
            when "1011" =>   --Noviembre
                  dias30 <= '1';
            when "1100" =>   --Diciembre
                  dias31 <= '1';
        end case;
    end process;
end architecture circuitoIF;
```

Código VHDL 1.9: Solución al Apartado 2.a: circuito combinacional descrito usando un bloque **process**.

El banco de pruebas del circuito combinacional se muestra en Código VHDL 1.12–1.13. Los dos cronogramas obtenidos al simular el banco de pruebas usando como circuito a testear los diseños de los apartados 2.a y 2.c se muestran, respectivamente, en las Figuras 1.5 y 1.6.

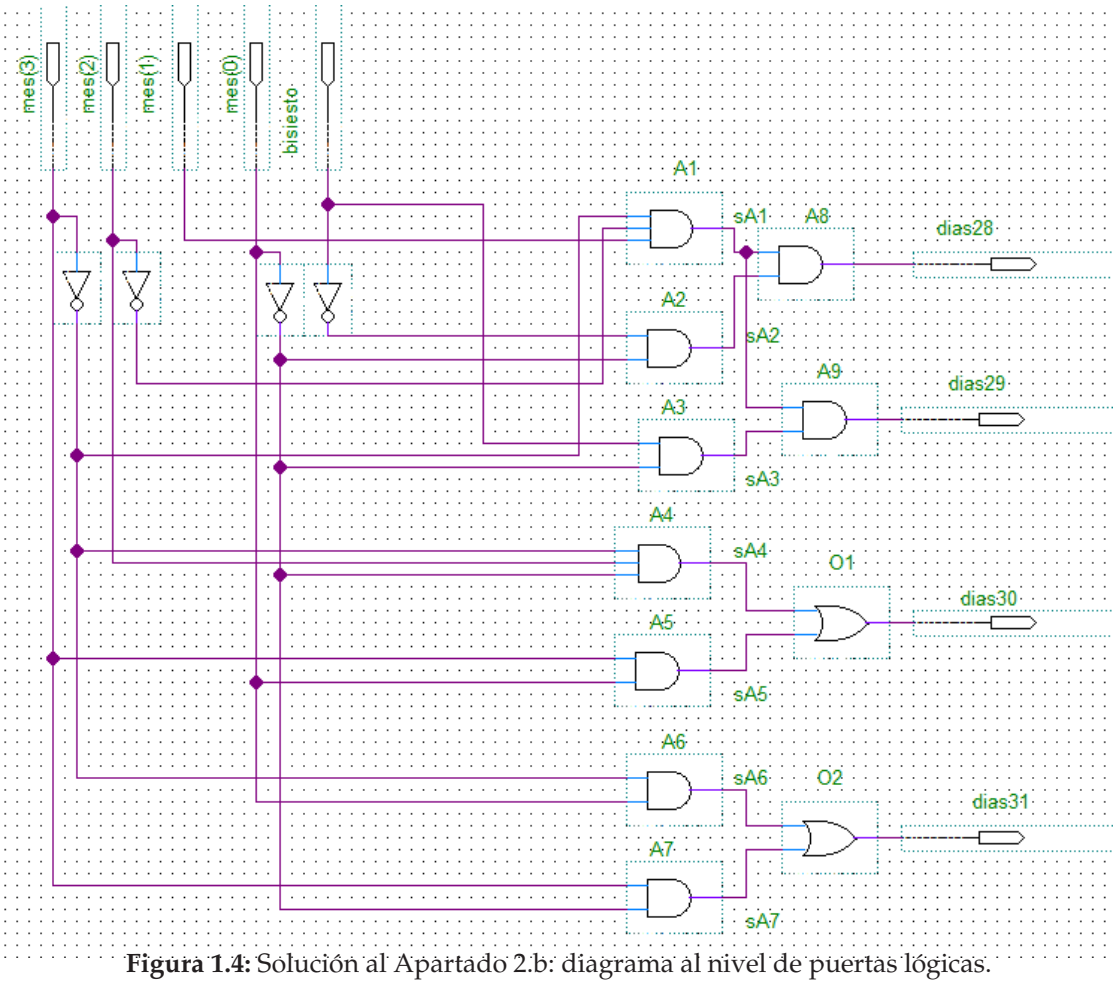

Figura 1.4: Solución al Apartado 2.b: diagrama al nivel de puertas lógicas.

```
-------------------------------------
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 1.10: Puerta lógica AND de 3 entradas.

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

architecture circuitoEST of circuito is
   signal m3n, m2n, m0n, bisieston: std_logic;
   signal sA1, sA2, sA3, sA4, sA5, sA6, sA7: 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 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 (m3n, mes(3));
   N2 : component not1 port map (m2n, mes(2));
   N3 : component not1 port map (m0n, mes(0));
   N4 : component not1 port map (bisieston, bisiesto);
   A1 : component and3 port map (sA1, m3n, m2n, mes(1));
   A2 : component and2 port map (sA2, m0n, bisieston);
   A3 : component and2 port map (sA3, m0n, bisiesto);
   A4 : component and3 port map (sA4, m3n, mes(2),m0n);
   A5 : component and2 port map (sA5, mes(3), mes(0));
   A6 : component and2 port map (sA6, m3n, mes(0));
   A7: component and2 port map (sA7, mes(3), m0n);
   A8 : component and2 port map (dias28, sA1, sA2);
   A9 : component and2 port map (dias29, sA1, sA3);
   O1 : component or2 port map (dias30, sA4, sA5);
   O2 : component or2 port map (dias31, sA6, sA7);
end architecture circuitoEST;
```

Código VHDL 1.11: Solución al Apartado 2.c: descripción estructural del circuito.

```
  ----------------------------------------------------
-- Banco de pruebas del circuito Ejercicio 2
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_circuito is
    constant MAX_COMB : integer := 16;      -- Num. combinaciones entrada
    constant DELAY    : time    := 10 ns; -- Retardo usado en el test
end entity bp_circuito;
architecture bp_circuito of bp_circuito is
    -- Salidas UUT
    signal dias28, dias29, dias30, dias31          : std_logic;
    -- Entradas UUT
    signal mes : std_logic_vector(3 downto 0);
    signal bisiesto      : std_logic;
  component circuito is
       port ( dias28, dias29, dias30, dias31: out std_logic;
           mes : in std_logic_vector(3 downto 0 );
           bisiesto: in std_logic);
    end component circuito;
begin   -- Cuerpo de la arquitectura
  UUT : component circuito port map
         (dias28, dias29, dias30, dias31, mes, bisiesto);
  main : process is
    variable temp : unsigned (3 downto 0); -- Usado en los cálculos
    variable esperado_dias : std_logic_vector(3 downto 0);
    variable error_count          : integer     := 0;
  begin
    report "Comienza la simulación";
    -- Generar todos los posibles valores de entrada
   for j in std_logic range '0' to '1' loop
      bisiesto <=std_logic(j);
      for i in 0 to (MAX_COMB-1) loop
          temp := TO_UNSIGNED(i,4);
          mes(3) <= std_logic(temp(3));
          mes(2) <= std_logic(temp(2));
          mes(1) <= std_logic(temp(1));
          mes(0) <= std_logic(temp(0));
          -- Calcular el valor esperado
          if      (i=1)or(i=3)or(i=5)or(i=7)or(i=8)or(i=10)or(i=12)          then
esperado_dias := "0001";
          elsif (i=2)and (j='0') then   esperado_dias := "1000";
          elsif (i=2)and(j='1') then   esperado_dias := "0100";
          elsif (i=4)or(i=6)or(i=9)or(i=11) then esperado_dias := "0010";
          else                    esperado_dias := "----";
          end if;
          wait for DELAY; -- Espera y compara con las salidas de UUT
          if ( esperado_dias /= dias28&dias29&dias30&dias31 and i>0
and i<13) then
```

Código VHDL 1.12: Solución al Apartado 2.d: banco de pruebas del circuito.

```
            report "ERROR en la salida valida. Valor esperado(dias28,
dias 29,dias30 y dias31:"  &
                   std_logic'image(esperado_dias(3))                &
                   std_logic'image(esperado_dias(2))                &
                   std_logic'image(esperado_dias(1))                &
                   std_logic'image(esperado_dias(0))                &

                   ",valor actual:"                                   &
                   std_logic'image(dias28)                          &
                   std_logic'image(dias29)                          &
                   std_logic'image(dias30)                          &
                   std_logic'image(dias31)                          &
                   " en el mes "                                        &
                   integer'image(i)                                  &

                   " en el instante:"                                   &
                   time'image(now);
             error_count := error_count + 1;
           end if;
        end loop;
    end loop; -- Final del bucle for de posibles valores de entrada
      -- Informe del número total de errores
      if (error_count = 0) then
           report "Simulación finalizada sin errores";
      else
           report   "ERROR: Hay "              &
                    integer'image(error_count) &
                    " errores.";
      end if;

      wait;      -- Final de la simulación
   end process main;
end architecture bp_circuito;
-------------------------------------------------
```

Código VHDL 1.13: Continuación del banco de pruebas del circuito (solución del apartado 2.d).

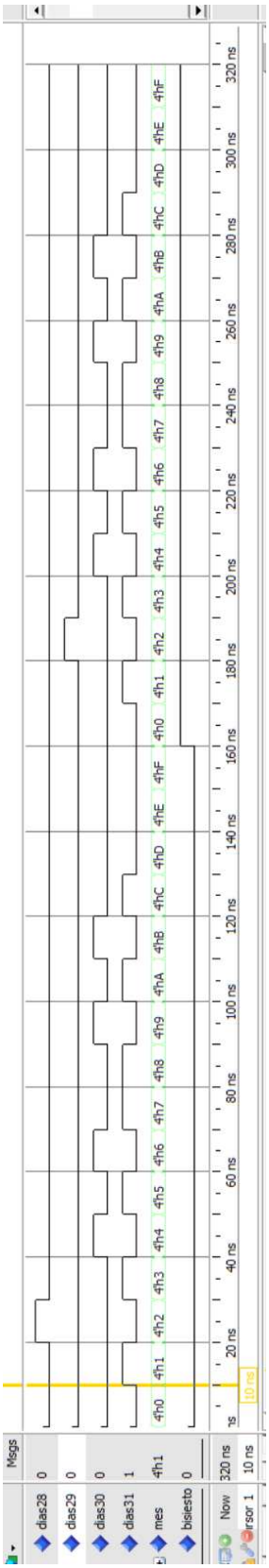

Figura 1.5: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.a.

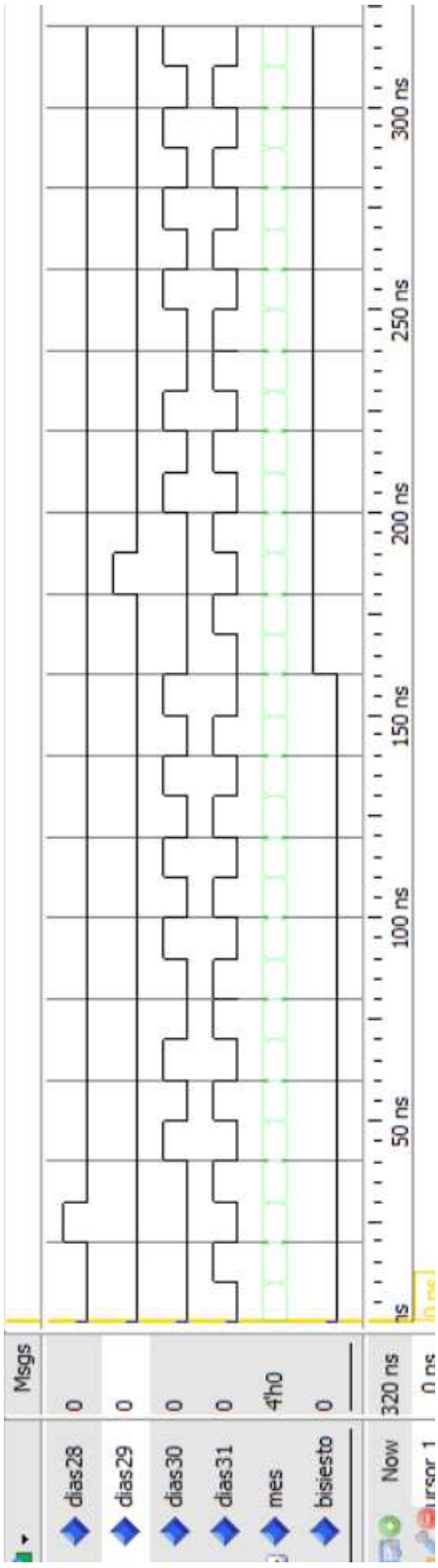

Figura 1.6: Cronograma del Apartado 2.d comprobando el comportamiento del circuito diseñado en el apartado 2.c.