INGENIERIA de COMPUTADORES III

APELLIDOS Y NOMBRE: **Periscal Porteiro, Juan**

IDENTIFICADOR: **jperiscal1**

DNI: **53306672D**

CENTRO ASOCIADO MATRICULADO: **047000 – A CORUÑA**

CENTRO ASOCIADO DE LA SESIÓN DE CONTROL: **047000 – A CORUÑA**

EMAIL DE CONTACTO: **jperiscal1@alumno.uned.es**

TELÉFONO DE CONTACTO: **625773978**

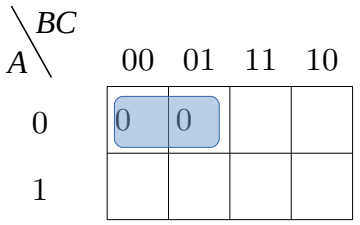
Ejercicio 1

A	B	C	F	G	
0	0	0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$
0	0	1	0	1	$\overline{A}\,\overline{B}\,C$
0	1	0	1	0	$\overline{A}\,B\,\overline{C}$
0	1	1	1	1	$\overline{A}\,BC$
1	0	0	1	0	$A\,\overline{B}\,\overline{C}$
1	0	1	1	1	$A\,\overline{B}\,C$
1	1	0	1	0	$A\,B\,\overline{C}$
1	1	1	1	0	$A\,B\,C$

$$F = (A+B+C)(A+B+\overline{C})$$
$$G = \overline{A}\,\overline{B}\,C + \overline{A}\,BC + A\,\overline{B}\,C$$

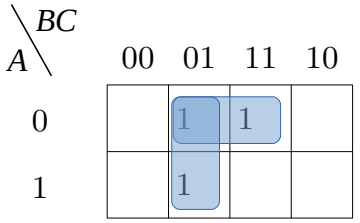
Minimización mediante mapas de Karnaugh

Minimización del producto de una suma para la señal de salida F



$F = A+B$

Minimización del producto de una suma para la señal de salida G



$$G = \overline{A}\,C + \overline{B}\,C$$
$$G = (\overline{A}+\overline{B})C$$

Apartado 1a: 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 .

Entity *circuito ejercicio1*

```
-- ================================
-- entity del circuito
-- fichero: fun1.vhd
library IEEE;
use IEEE.std_logic_1164.all;

entity fun1 is port
    (F,G    : out std_logic;
     A,B,C  : in std_logic);
end entity fun1;
--================================
```

Apartado 1b: Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

Descripción *comportamiento del circuito del ejercicio 2*

```
-- ================================
-- Comportamiento circuito
--fichero: archo_comp_fun1.vhd

library IEEE;
use IEEE.std_logic_1164.all;

architecture arch_comp_fun1 of fun1 is
begin
F <= A or B;
G <= (not(A) or not(B)) and C;
end architecture arch_comp_fun1;
-- ================================
```

Apartado 1c: 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.

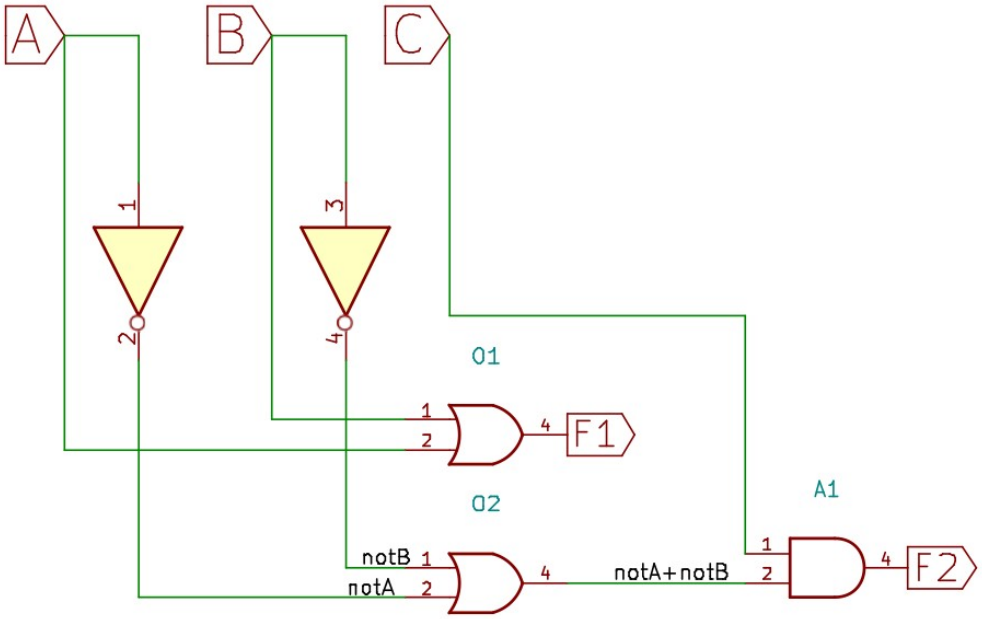


***Diagrama** del circuito del ejercicio 1*

Puerta AND

```
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Puerta AND de 2 entradas
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;
-- ================================
```

Puerta NOT

```
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Inversor de 1 entrada
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;
-- ================================
```

Puerta OR

```
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Puerta OR de 2 entradas
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;
-- ================================
```

Apartado 1d: Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

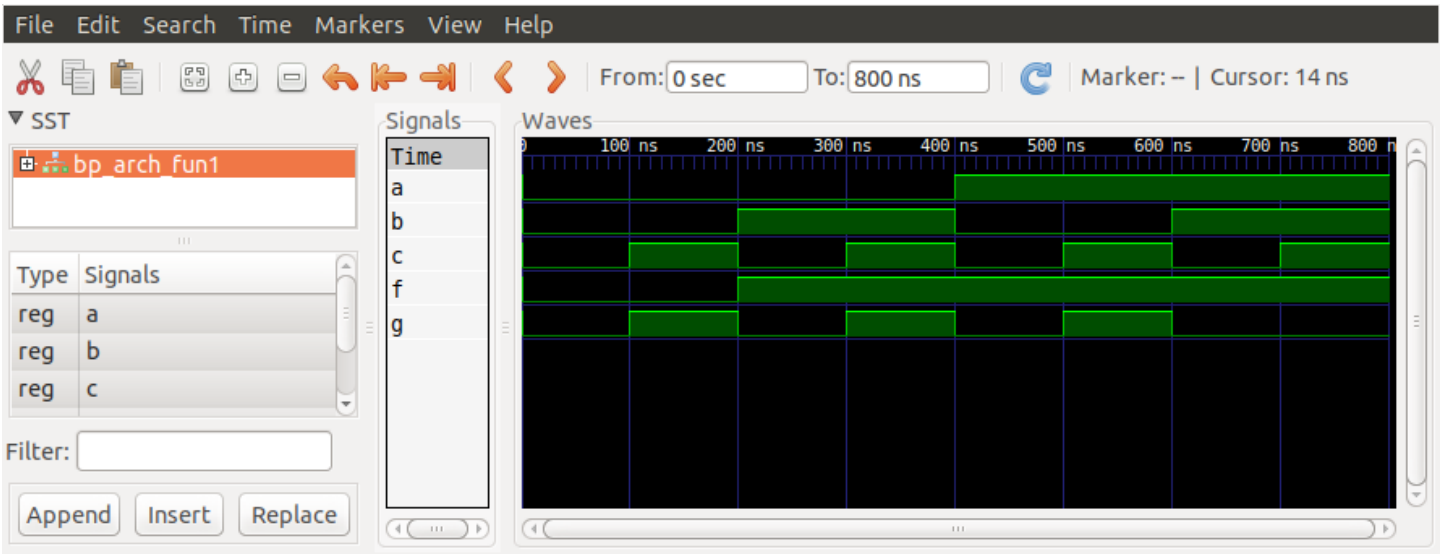
Descripción *estructura del circuito del ejercicio 1*

```
-- ================================
-- Descripcion estructura del circuito
-- fichero: arch_estructura_fun1.vhd

library IEEE;
use IEEE.std_logic_1164.all;

architecture arch_estructura_fun1 of fun1 is
    signal notA         : std_logic;
    signal notB         : std_logic;
    signal notA_or_notB : std_logic;

    -- Declaración de las clases de componentes
    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;

    component and2 is port
        (y0     : out std_logic;
         x0,x1  : in std_logic);
    end component and2;

begin
    -- Instanciación y coneción de los componentes
    N1: component not1 port map(y0 => notA, x0 => A);
    N2: component not1 port map(y0 => notB, x0 => B);
    O1: component or2 port map(y0 => notA_or_notB, x0 => notA, x1 => notB);
    O2: component or2 port map(y0 => F1, x0 => A, x1 => B);
    A1: component and2 port map(y0 => F2, x0 => C, x1 => notA_or_notB);

end architecture arch_estructura_fun1;
-- ================================
```

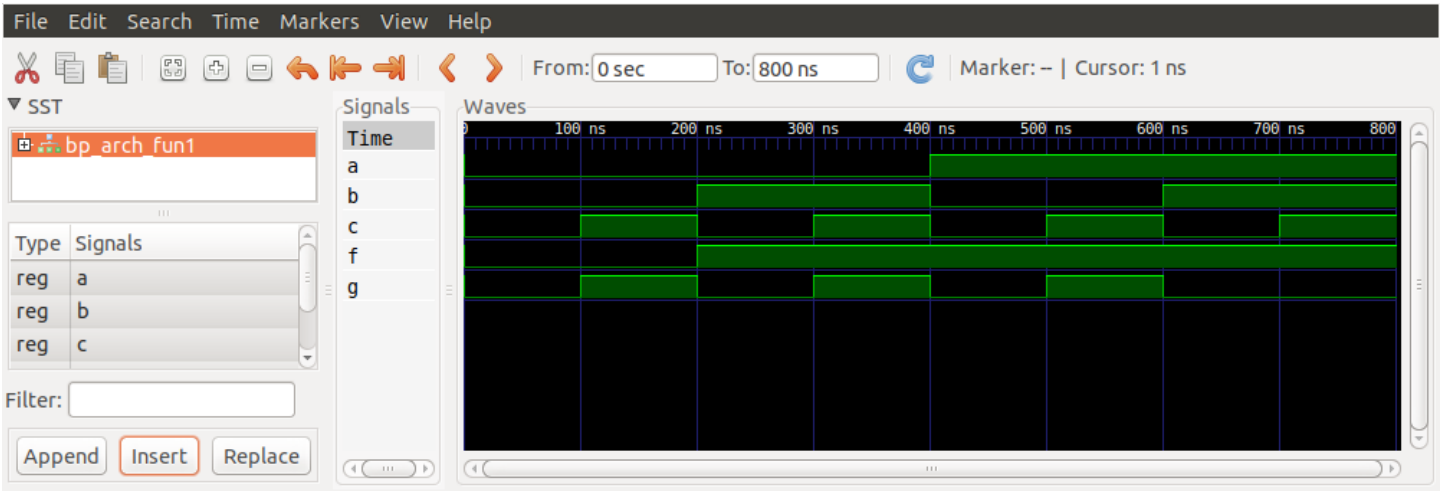
Apartado 1e: Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, la salida del circuito cuya **entity** ha especificado en el Apartado 1.a. Emplee dicho banco de pruebas para comprobar mediante inspección visual que los dos diseños de los Apartado 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.

Banco de pruebas *del circuito Ejercicio 1*

```
-- ================================
-- Banco de pruebas del circuito Ejercicio 1
-- fichero: bp_arch_fun1.vhd
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_arch_fun1 is
constant DELAY : time:= 100 ns; -- Retardo usado en el test
end entity bp_arch_fun1;

architecture bp_arch_fun1 of bp_arch_fun1 is

    signal F, G : std_logic;
    signal A, B, C  : std_logic;

    component fun1 is port
        (F,G    : out std_logic;
        A,B,C   : in std_logic);
    end component fun1;

begin
    -- Instancia del circuito que va a ser testeado
    UUT: component fun1 port map (F,G,A,B,C);

    -- Vectores del test
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;
        -- Final de la simulación
        wait;
    end process vec_test;
end architecture bp_arch_fun1;
-- ================================
```



***Cronograma**. Comportamiento circuito ejercicio 1*



***Cronograma**. Comportamiento circuito ejercicio 2*

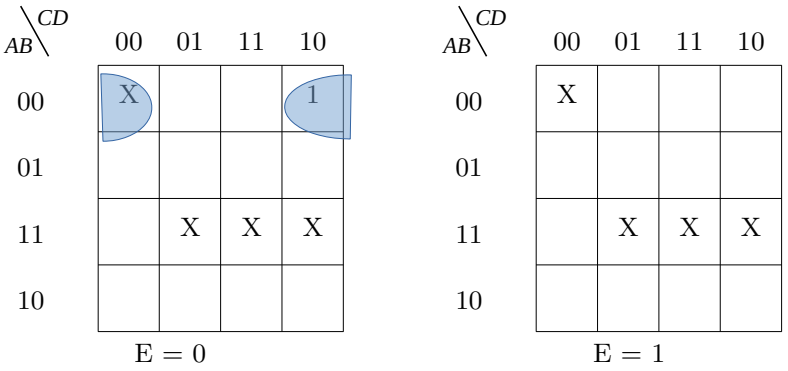
Ejercicio 2

mes				bisiesto	dias			
					28	29	30	31
0	0	0	0	0	X	X	X	X
0	0	0	1	0				1
0	0	1	0	0	1			
0	0	1	1	0				1
0	1	0	0	0			1	
0	1	0	1	0				1
0	1	1	0	0			1	
0	1	1	1	0				1
1	0	0	0	0				1
1	0	0	1	0			1	
1	0	1	0	0				1
1	0	1	1	0			1	
1	1	0	0	0				1
1	1	0	1	0	X	X	X	X
1	1	1	0	0	X	X	X	X
1	1	1	1	0	X	X	X	X
0	0	0	0	1	X	X	X	X
0	0	0	1	1				1
0	0	1	0	1		1		
0	0	1	1	1				1
0	1	0	0	1			1	
0	1	0	1	1				1
0	1	1	0	1			1	
0	1	1	1	1				1
1	0	0	0	1				1
1	0	0	1	1			1	
1	0	1	0	1				1
1	0	1	1	1			1	
1	1	0	0	1				1
1	1	0	1	1	X	X	X	X
1	1	1	0	1	X	X	X	X
1	1	1	1	1	X	X	X	X

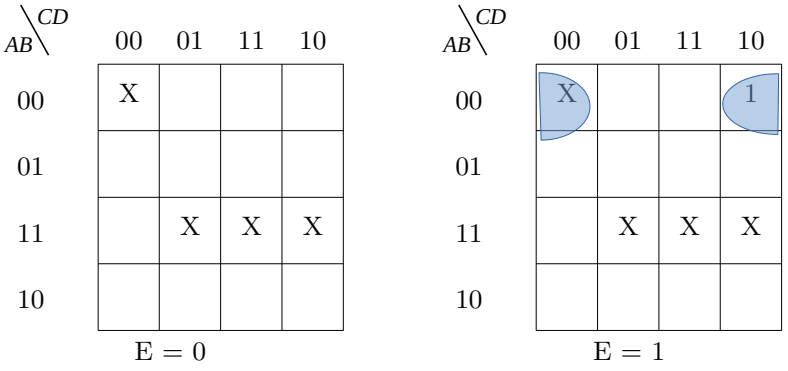
$F28 = \overline{A}\,\overline{B}\,\overline{D}\,\overline{E}$
$F29 = \overline{A}\,\overline{B}\,\overline{D}\,E$
$F30 = AD + \overline{A}\,B\,\overline{D}$
$F31 = \overline{A}\,D + A\,\overline{D} = A \oplus D$

Minimización mediante Karnaugh

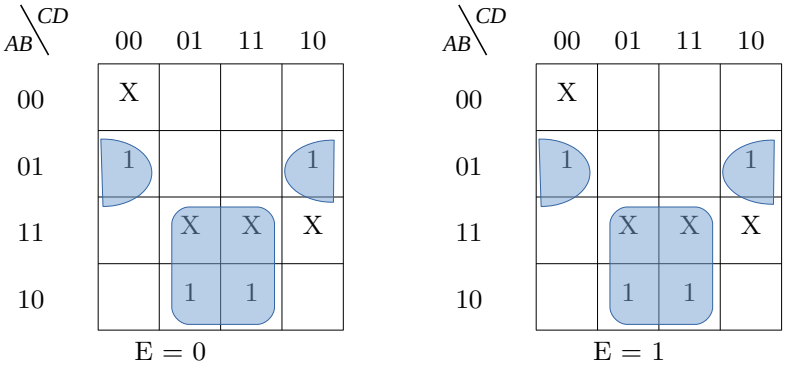
Minimización del producto de una suma para la señal de salida F1



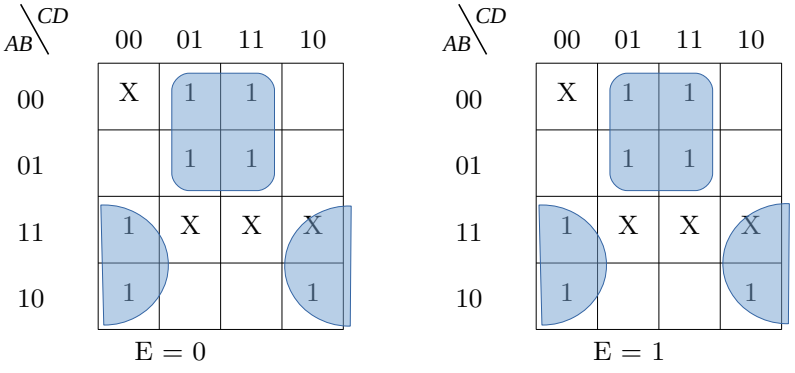
$F28 = \overline{A}\,\overline{B}\,\overline{D}\,\overline{E}$



$F29 = \overline{A}\,\overline{B}\,\overline{D}\,E$



$F30 = AD + \overline{A}\,B\,\overline{D}$



$F31 = \overline{A}\,D + A\,\overline{D} = A \oplus D$

Apartado 2a: 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.

Entity *del circuito del ejercicio 2*

```
-- ================================
-- entity del circuito del ejercicio2
--fichero: fun2.vhd

library IEEE;
use IEEE.std_logic_1164.all;

entity fun2 is
 port( F28,F29,F30,F31 : out std_logic;
 A,B,C,D,E : in std_logic);
end entity fun2;
-- ================================
```

Descripción **comportamiento** *circuito ejercicio 2*

```
-- ================================
-- Comportamiento circuito ejercicio 2
-- fichero: arch_comp_fun2.vhd

library IEEE;
use IEEE.std_logic_1164.all;

architecture arch_comp_fun2 of fun2 is
begin
process(A,B,C,D,E) is
 variable mes : std_logic_vector (3 downto 0);
 variable bisiesto : std_logic ;
 begin
 mes := A & B & C & D;
 bisiesto := E;
case mes is
-- Enero, Marzo, Mayo, Julio, Agosto, Octubre, Diciembre tienen 31 dias
when "0001"|"0011"|"0101"|"0111"|"1000"|"1010"|"1100" =>
F28 <= '0'; F29 <= '0'; F30 <= '0'; F31 <= '1';

-- Abril, Junio, Septiembre, Noviembre tienen 30 dias
when "0100"|"0110"|"1001"|"1011" =>
 F28 <= '0'; F29 <= '0'; F30 <= '1'; F31 <= '0';

-- Febrero
when "0010" =>
 F30 <= '0'; F31 <= '0';
 if(E='0') then F28 <= '1'; F29 <= '0'; -- Si no es bisiesto
 elsif (E='1') then F28 <= '0'; F29 <= '1'; -- Si es bisiesto
 else F28 <= 'X'; F29 <= 'X';
 end if;
when others =>
 F28 <= 'X'; F29 <= 'X'; F30 <= 'X'; F31 <= 'X';
end case;
end process;
end architecture arch_comp_fun2;
-- ================================
```

Apartado 2b: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.

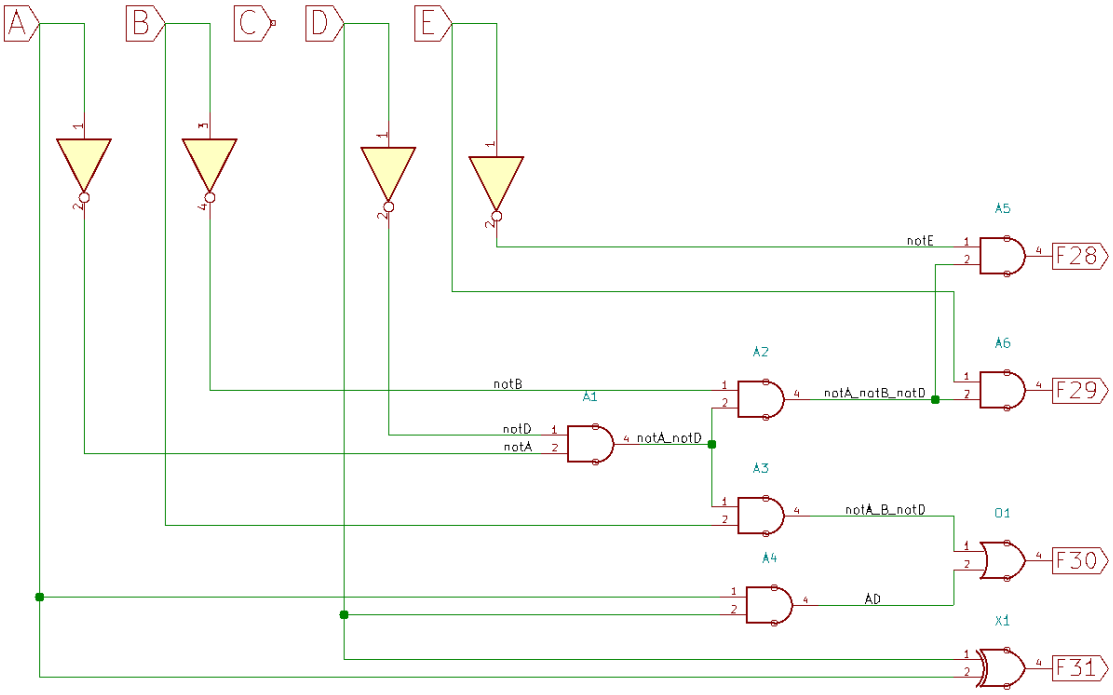


***Diagrama** del circuito del ejercicio 2*

Puerta AND

```vhdl
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Puerta AND de 2 entradas
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;
-- ================================
```

Puerta NOT

```vhdl
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Inversor de 1 entrada
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;
-- ================================
```

Puerta OR

```vhdl
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Puerta OR de 2 entradas
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;
-- ================================
```

Puerta XOR

```vhdl
-- ================================
library IEEE;
use IEEE.std_logic_1164.all;
-- Puerta OR de 2 entradas
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;
-- ================================
```

Apartado 2c: Escriba en VHDL una **architecture** que describa la estructura del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

Descripción ***estructura*** *del circuito del ejercicio 2*

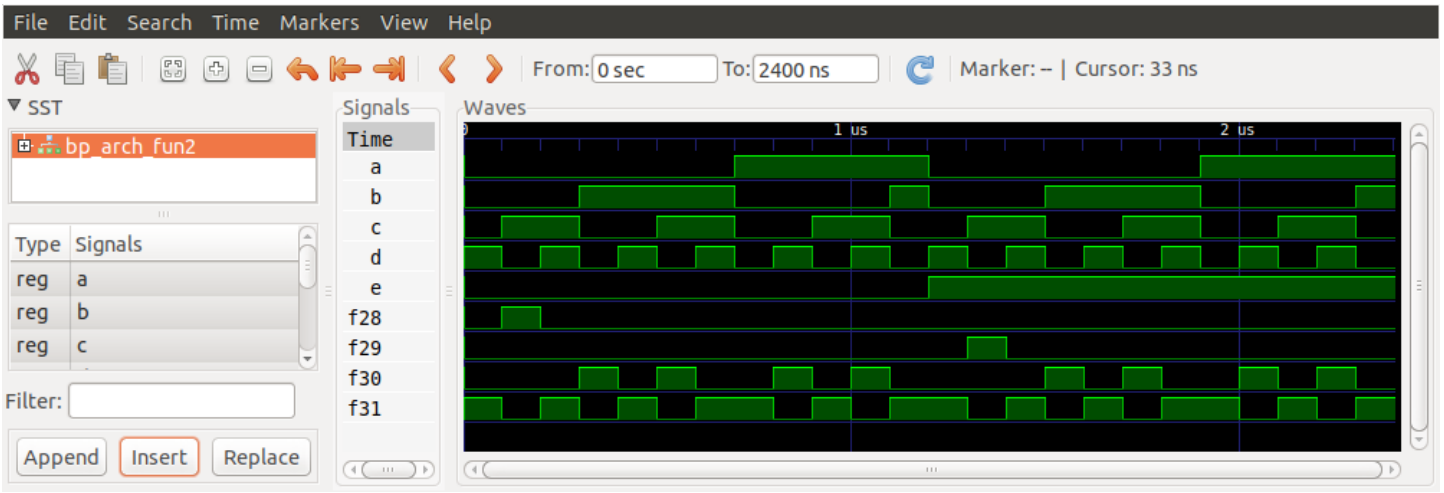
```
-- ================================
-- Estructura del circuito del ejercicio 2
-- fichero: arch_estructura_fun2.vhd

library IEEE;
use IEEE.std_logic_1164.all;

architecture arch_estructura_fun2 of fun2 is
signal notA,notB,notD,not_E : std_logic;
signal notA_notD :std_logic;
signal notA_notB_notD :std_logic;
    signal notA_B_notD :std_logic;
    signal AD :std_logic;

-- Declaración de las clases de componentes
    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;

    component and2 is port
        (y0     : out std_logic;
         x0,x1  : in std_logic);
    end component and2;

    component xor2 is port
        (y0     : out std_logic;
         x0,x1  : in std_logic);
    end component xor2;
begin
    -- Instanciación y conexión de los componentes
    N1: component not1 port map(x0 => A, y0 => notA);
    N2: component not1 port map(x0 => B, y0 => notB);
    N3: component not1 port map(x0 => D, y0 => notD);
    N4: component not1 port map(x0 => E, y0 => not_E);
    A1: component and2 port map(x0 => notA, x1 => notD, y0 => notA_notD);
    A2: component and2 port map(x0 => notB, x1 => notA_notD, y0 => notA_notB_notD);
    A3: component and2 port map(x0 => notA_notD, x1 => B, y0 => notA_B_notD);
    A4: component and2 port map(x0 => A, x1 => D, y0 => AD);
    A5: component and2 port map(x0 => not_E,x1 => notA_notB_notD, y0 => F28);
    A6: component and2 port map(x0 => E, x1 => notA_notB_notD, y0 => F29);
    O1: component or2 port map(x0 => AD,x1 => notA_B_notD, y0 => F30);
    x1: component xor2 port map(x0 => A, x1 => D, y0 => F31);

end architecture arch_estructura_fun2;
-- ================================
```

Apartado 2d: 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.

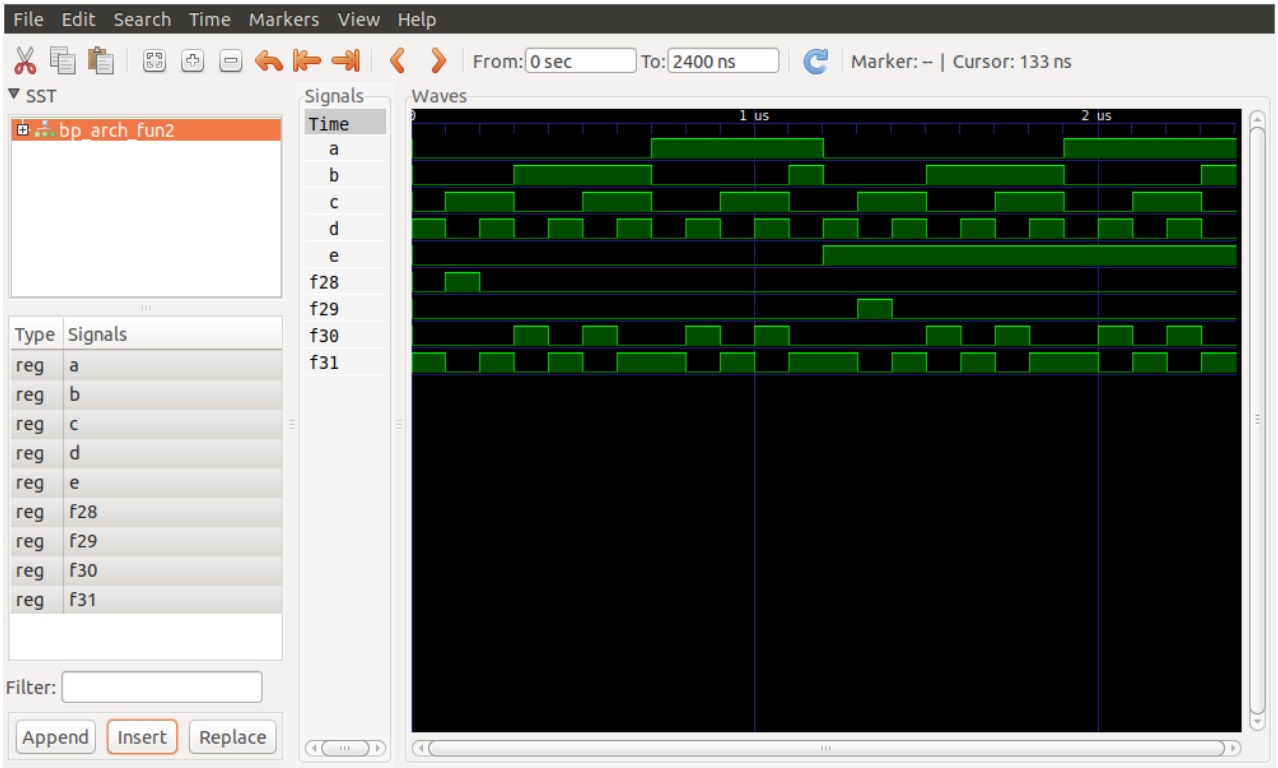
Banco de pruebas *del circuito Ejercicio 2*

```
-- ================================
-- Banco de pruebas del circuito Ejercicio 2
-- fichero: bp_arch_fun2.vhd

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

entity bp_arch_fun2 is
 constant DELAY : time:= 100 ns; -- Retardo usado en el test
end entity bp_arch_fun2;

architecture bp_arch_fun2 of bp_arch_fun2 is
 signal F28,F29,F30,F31 : std_logic;
 signal A,B,C,D,E : std_logic;

 component fun2 is port
 (F28,F29,F30,F31 : out std_logic;
 A,B,C,D,E : in std_logic);
 end component fun2;

begin
-- Instancia del circuito que va a ser testeado
UUT : component fun2 port map(F28,F29,F30,F31,A,B,C,D,E);
-- Vectores del test
vec_test : process is
       variable mes : std_logic_vector (3 downto 0);
       variable salidas : std_logic_vector (3 downto 0);
       variable bisiesto : std_logic;
       variable exito : boolean;
       variable dias : std_logic_vector (3 downto 0);

       begin
       exito := true;
       -- Generar todos los posibles valores de entrada
       for j in std_logic range '0' to '1' loop
         bisiesto :=j;
         E <= bisiesto;
             for i in 1 to 12 loop
             mes := std_logic_vector(to_unsigned(i,4));
             A <= mes(3);
             B <= mes(2);
             C <= mes(1);
             D <= mes(0);

             salidas := F28 & F29 & F30 & F31;
       case mes is
       -- Enero, Marzo, Mayo, Julio, Agosto, Octubre, Diciembre tienen 31 salidas
       when "0001"|"0011"|"0101"|"0111"|"1000"|"1010"|"1100" =>
             dias := "0001"; -- 31 dias

       -- Abril, Junio, Septiembre, Noviembre tienen 30 salidas
       when "0100"|"0110"|"1001"|"1011" =>
             dias := "0010"; -- 30 dias

       -- Febrero
```

```
            when "0010" =>
                  if (E='0') then dias := "1000"; -- 28 dias
                  elsif (E='1') then dias := "0100"; --29 dias
                  end if;
            when others => dias := "XXXX";
            end case;
            -- Una vez el numerdo de dias no coincide con la salida 'exito' en 'false'
                  if(dias /= salidas) then exito := (exito and false); end if;
                  wait for DELAY;
            end loop;
      end loop;
            assert exito report "Test no superado." severity note;
            -- Final de la simulación
            wait;
      end process vec_test;
    end architecture bp_arch_fun2;
```



Cronograma. *Comportamiento circuito ejercicio 2*



Cronograma. *Estructura circuito del ejercicio 2*