INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2022

EJERCICIO 1

Dadas las dos funciones lógicas (`F` y `G`) mostradas a continuación, que dependen de 3 variables (`x`, `y` y `z`):

```
F = (x and not(y) and not(z)) or (x and y and not(z))
G = (not(x) and y) or (x and not(y))
```

1.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito.

1.b) (1 punto) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

1.c) (0.5 puntos) Simplifique las funciones lógicas y dibuje el diagrama de un circuito que implemente estas dos funciones lógicas ya simplificadas al nivel de puertas lógicas. Para ello, se ha de emplear únicamente puertas AND de dos entradas, puertas NOT y puertas XOR de dos entradas. A continuación, escriba en VHDL la **entity** y la **architecture** de cada una de las puertas lógicas AND de dos entradas, NOT y XOR de dos entradas que ha empleado en el diseño del anterior circuito.

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) (1 punto) Escriba en VHDL un banco de pruebas que permita visualizar, para todos los posibles valores de las entradas, las salidas de los circuitos diseñados en los Apartados 1.b y 1.d. Compruebe mediante inspección visual que los dos diseños funcionan correctamente. Incluya en la memoria los dos cronogramas obtenidos al realizar la simulación del banco de pruebas del circuito diseñado en los Apartados 1.b y 1.d.

Solución al Ejercicio 1

La **entity** del circuito se muestra en Código VHDL 1.1.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
entity funcLog_F_G is
   port ( F, G     : out std_logic;
          x, y, z : in std_logic );
end entity funcLog_F_G;
-------------------------------------
```

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

El Código VHDL 1.2 muestra la **architecture** del circuito describiendo su comportamiento.

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

architecture funcLog_F_G_Comp of funcLog_F_G is
begin
    F <= (x and (not y) and (not z)) or ( x and y and (not z));
    G <= ( (not x) and y) or ( x    and ( not y));
end architecture funcLog_F_G_Comp;
-------------------------------------
```

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

La Figura 1.1 muestra el diagrama del circuito empleando una puerta AND, un inversor y una puerta XOR.

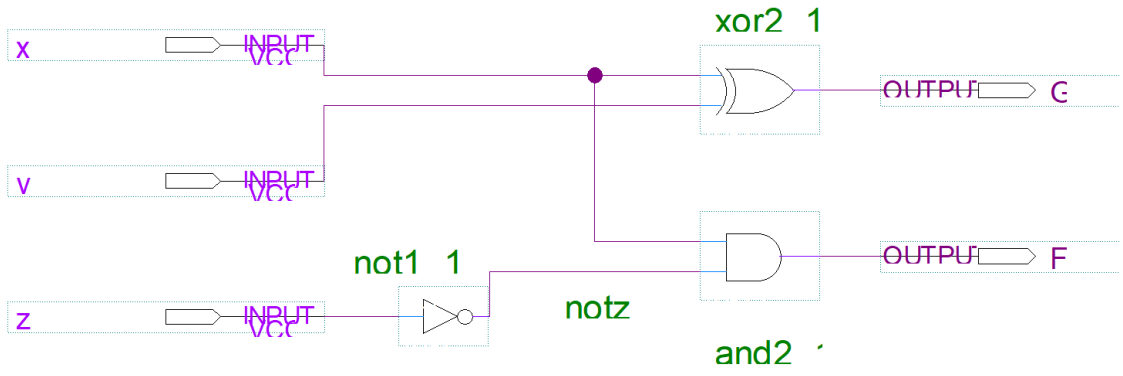

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

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, respectivamene.

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

```
-----------------------------------
-- OR exclusiva de 2 entradas: xor2
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 1.6 muestra la **architecture** del circuito describiendo estructura.

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.7.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture funcLog_F_G_Estruc of funcLog_F_G is
   signal notz : 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 xor2 is
      port ( y0 : out std_logic;
          x0, x1 : in std_logic );
   end component xor2;
begin
-- Instanciación y conexión de los componentes
   not1_1 : component not1 port map (notz, z);
   and2_1 : component and2 port map (F, notz, x);
   xor2_1 : component xor2 port map (G, x, y);
end architecture funcLog_F_G_Estruc;
-----------------------------------
```

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

Los dos cronogramas obtenidos al simular el banco de pruebas con los diseños obtenidos en los Apartados 1.b y 1.d se muestran, respectivamente, en las Figuras 1.2 y 1.3.

```
----------------------------------
-- Banco de pruebas
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_funcLog_F_G is
end entity bp_funcLog_F_G;

architecture bp_funcLog_F_G of bp_funcLog_F_G is
   signal F, G : std_logic; -- Conectar salidas UUT
   signal x0, x1, x2 : std_logic; -- Conectar entradas UUT

   component funcLog_F_G is port
    ( F, G : out std_logic;
      x, y, z: in std_logic);
   end component funcLog_F_G;

begin
-- Instanciar y conectar UUT
   uut : component funcLog_F_G port map
   ( F => F, G => G,
   x => x0, y => x1, z => x2);

   gen_vec_test : process
     variable test_in : unsigned (2 downto 0); -- Vector de test
   begin
     test_in := B"000";
     for count in 0 to 7 loop
       x2 <= test_in(2);
       x1 <= test_in(1);
       x0 <= test_in(0);
       wait for 10 ns;
       test_in := test_in + 1;
     end loop;
     wait;
   end process gen_vec_test;
end architecture bp_funcLog_F_G;
----------------------------------
```

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

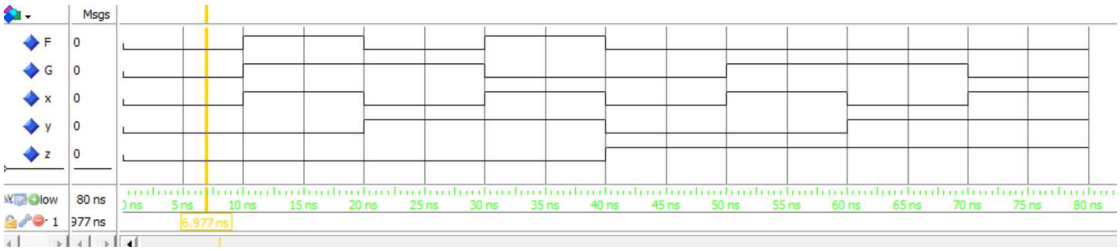

Figura 1.2: Cronograma del Apartado 1.e comprobando el comportamiento del circuito diseñado en el Apartado 1.b.

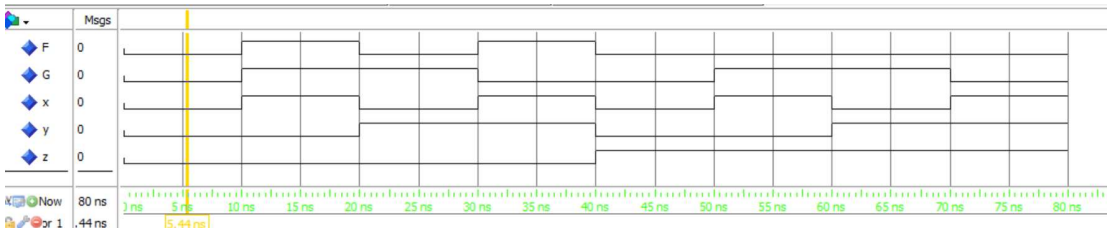

Figura 1.3: Cronograma del Apartado 1.e comprobando el comportamiento del circuito diseñado en el Apartado 1.d.

EJERCICIO 2

Se quiere diseñar un circuito secuencial síncrono que opera en el flanco de subida de la señal de reloj cuyo propósito es el control del sistema de alarma de un coche que enciende o apaga una sirena.

El circuito tiene las cinco señales de entrada de un bit siguientes:

- Señal `M` que permite controlar la alarma según su valor sea '1' o '0'.
- Señal `D`, que indica si la puerta del coche está abierta o cerrada. Si `D` vale '1' la puerta está abierta, y si vale '0' está cerrada.
- Señal `V`, que indica si existe una vibración en el coche. Si `V` vale '1' existen vibraciones, y si vale '0' no existen vibraciones.
- Señal de reloj llamada `clk`.
- Señal de reset asíncrona activa a nivel alto llamada `reset`.

El circuito tiene una única señal de salida llamada `S` que indica si la sirena del coche está encendida o apagada. Si la señal `S` vale '1' la sirena está encendida, y si vale '0' está apagada.

El circuito se ha de diseñar como una máquina de estados de Moore con los dos estados siguientes:

- Estado ON. En este estado la sirena está encendida. Una vez el circuito entra en este estado ON, el circuito permanece en el estado ON mientras la señal `M` tenga el valor '1', con independencia del valor de las señales `D` y `V`. El circuito pasa del estado ON al estado OFF cuando la señal `M` tiene el valor '0'.
- Estado OFF. En este estado la sirena está apagada. El circuito pasa del estado OFF al estado ON solo si la señal `M` tiene el valor '1' y, además, la señal `D` tiene el valor '1' o la señal `V` tiene el valor '1' o ambas tienen el valor '1'.

Cuando la señal `reset` tiene valor '1' el circuito pasa asíncronamente al estado OFF. El resto de transiciones entre estados son síncronas, teniendo siempre lugar en el flanco de subida de la señal de reloj.

2.a (3 puntos) Proporcione el código VHDL de la **architecture** del circuito secuencial anterior describiendo su comportamiento como una máquina de estados de Moore. En la memoria ha de mostrar, además de todo el código VHDL diseñado, el diagrama de estados del circuito. El diagrama de estados debe mostrar los dos estados y todas las posibles transiciones entre estados para cada uno de los posibles valores de las señales `M`, `D` y `V`.

La **entity** del circuito se muestra a continuación.

```
entity alarma is
   port (  S        : out std_logic;
           M, D, V  : in std_logic;
           clk      : in std_logic;
           reset    : in  std_logic);
end entity alarma;
```

2.b (3 puntos) Programe en VHDL un banco de pruebas que testee el circuito que ha diseñado en el Apartado 2.a. El banco de pruebas debe comparar la salida de la UUT con la salida esperada, mostrando el correspondiente mensaje de error en caso de que la salida obtenida de la UUT no corresponda con la esperada. En el programa de test todos los arcos de las transiciones entre estados han de ser recorridos por los menos una vez. Incluya en la memoria el cronograma obtenido al realizar la simulación del banco de pruebas del circuito diseñado en el Apartados 2.a y el diagrama de estados del circuito. En el diagrama de estados se debe mostrar junto a cada flecha de la transición de estado un número que refleje el orden en que se ha recorrido dicha transición de estado.

Solución al Ejercicio 2

La Figura 1.4 muestra el diagrama diagrama de estados del circuito.

El Código VHDL 1.8 muestra el diseño del circuito describiendo su comportamiento.

La Figura 1.5 muestra el orden en que en banco de pruebas recorre cada transición de estados. El Código VHDL 1.9–1.10 muestra el código VHDL banco de pruebas del circuito. El cronograma obtenido al simular el banco de pruebas usando como circuito a testear el diseño del Apartado 2.a se muestra en la Figura 1.6.

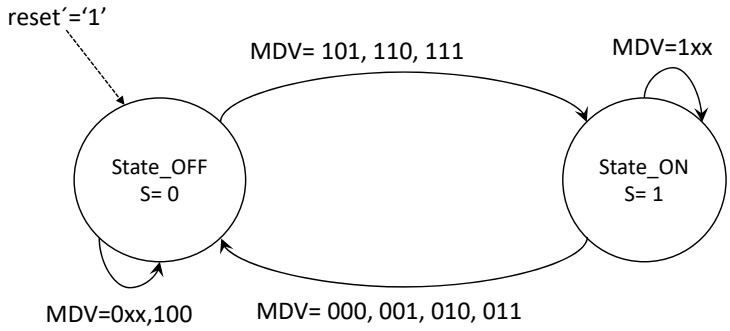

Figura 1.4: Solución al Apartado 2.a: diagrama de estados del circuito.

```
-------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture alarmaComp of alarma is
   constant state_ON: std_logic := '1';
   constant state_OFF: std_logic := '0';
   signal state: std_logic;
begin
    process (clk, reset) is

    begin
        if (reset = '1') then
            state <= state_OFF;
        elsif (rising_edge(clk)) then
            case state is
                when state_OFF =>
                   if ((M='1') and (V='1' or D='1')) then
                       state <= state_ON;
                   end if;
                when state_ON =>
                   if (M='0') then
                       state <= state_OFF;
                   end if;
                when others=>
            end case;
        end if;
    end process;
    S <= '1' when (state = state_ON) else '0';
end architecture alarmaComp;
-------------------------------------
```

Código VHDL 1.8: Solución al Apartado 2.a: circuito alarma.

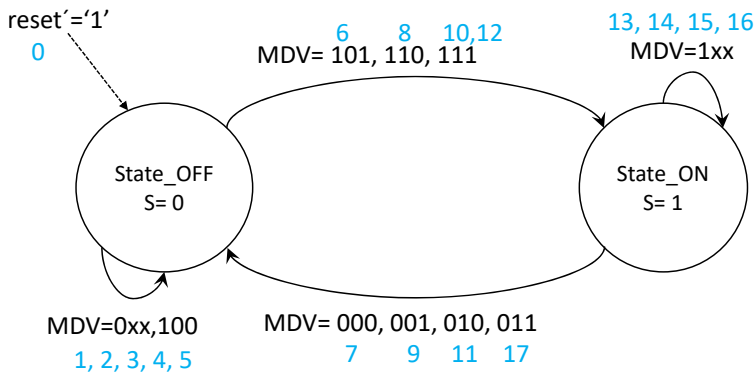

Figura 1.5: Solución al Apartado 2.b: diagrama de estados del circuito indicando el orden en que se recorren las transiciones.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity bp_alarma is
end entity bp_alarma;
architecture bp_alarma of bp_alarma is
   constant PERIODO    : time          := 20 ns; -- Reloj
   signal     S       : std_logic;     -- Salida UUT
   signal     M, D, V            : std_logic; -- Entradas UUT

   signal     clk         : std_logic   := '0';
   signal     reset   : std_logic;
   component alarma is
    port(            S    : out std_logic;
              M, D, V   : in std_logic;
              clk         : in std_logic;
              reset       : in    std_logic);
   end component alarma;
    -- Procedimiento para comprobar las salidas
    procedure comprueba_salidas
        (esperado_S       :         std_logic;
          actual_S         :          std_logic;
          error_count      :          inout integer) is
    begin
          -- Comprueba state

         if (esperado_S /= actual_S ) then
             report "ERROR: Salida Y esperada ("      &
                     std_logic'image(esperado_S)     &
                     "), salida actual ("             &
                     std_logic'image(actual_S)       &
                     "), instante: "                  &
                     time'image(now);
             error_count := error_count + 1;
         end if;
     end procedure comprueba_salidas;
begin
   -- Instanciar y conectar UUT
   uut : component alarma port map
                    (S, M, D, V, clk, reset);

   reset <=   '1','0' after (PERIODO/2+PERIODO/4);
   clk <= not clk after (PERIODO/2);
   gen_vec_test : process is
     variable error_count : integer := 0; -- Núm. errores
```

Código VHDL 1.9: Solución al Apartado 2.b: banco de pruebas del circuito (1/2).

```
   begin
      report "Comienza la simulación";
-- Vectores de test y comprobación del resultado
        M <= '0';D <= '0';V <= '0';    wait for 2*PERIODO;    -- 1
        comprueba_salidas('0',S,error_count);
        M <= '0';D <= '0';V <= '1';    wait for PERIODO;    -- 2
         comprueba_salidas('0',S,error_count);
        M <= '0';D <= '1';V <= '0';    wait for PERIODO;    -- 3
        comprueba_salidas('0',S,error_count);
        M <= '0';D <= '1';V <= '1';    wait for PERIODO;    -- 4
        comprueba_salidas('0',S,error_count);
        M <= '1';D <= '0';V <= '0';    wait for PERIODO;    -- 5
        comprueba_salidas('0',S,error_count);
        M <= '1';D <= '0';V <= '1';    wait for PERIODO;    -- 6
        comprueba_salidas('1',S,error_count);
        M <= '0';D <= '0';V <= '0';    wait for PERIODO;    -- 7
        comprueba_salidas('0',S,error_count);
        M <= '1';D <= '1';V <= '0';    wait for PERIODO;    -- 8
        comprueba_salidas('1',S,error_count);
        M <= '0';D <= '0';V <= '1';    wait for PERIODO;    -- 9
        comprueba_salidas('0',S,error_count);
        M <= '1';D <= '1';V <= '1';    wait for PERIODO;    -- 10
        comprueba_salidas('1',S,error_count);
         M <= '0';D <= '1';V <= '0';    wait for PERIODO;    -- 11
        comprueba_salidas('0',S,error_count);
         M <= '1';D <= '1';V <= '1';    wait for PERIODO;    -- 12
        comprueba_salidas('1',S,error_count);
        M <= '1';D <= '0';V <= '0';    wait for PERIODO;    -- 13
        comprueba_salidas('1',S,error_count);
        M <= '1';D <= '0';V <= '1';    wait for PERIODO;    -- 14
         comprueba_salidas('1',S,error_count);
        M <= '1';D <= '1';V <= '0';    wait for PERIODO;    -- 15
        comprueba_salidas('1',S,error_count);
        M <= '1';D <= '1';V <= '1';    wait for PERIODO;    -- 16
        comprueba_salidas('1',S,error_count);
        M <= '0';D <= '1';V <= '1';    wait for PERIODO;    -- 17
        comprueba_salidas('0',S,error_count);
      -- Informe final
        report   "Hay "                &
                 integer'image(error_count ) &
                 " errores.";
     wait;        -- Final del bloque process
   end process gen_vec_test;
end architecture bp_alarma;
-------------------------------------
```

Código VHDL 1.10: Solución al Apartado 2.b: banco de pruebas del circuito (2/2).

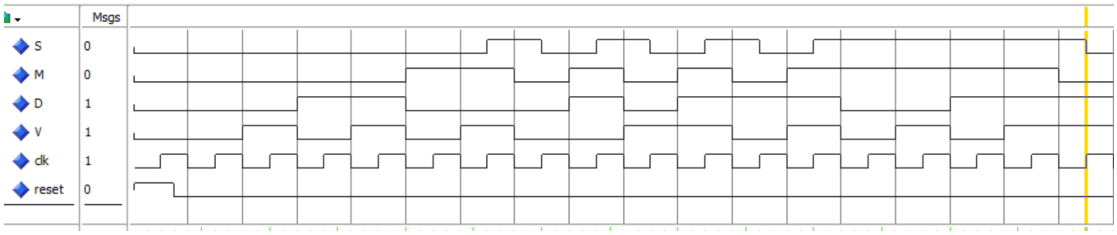

Figura 1.6: Cronograma del Apartado 2.b comprobando el comportamiento del circuito diseñado en el Apartado 2.a.