INGENIERÍA DE COMPUTADORES 3

Solución al Trabajo Práctico - Septiembre de 2017

EJERCICIO 1 (3 PUNTOS)

Se desea diseñar un circuito digital que implemente la función `F` cuya tabla de verdad se muestra a continuación, que depende de las tres variables `x`, `y` y `z`:

x	y	z	F
'0'	'0'	'0'	'1'
'0'	'0'	'1'	'1'
'0'	'1'	'0'	'0'
'0'	'1'	'1'	'0'
'1'	'0'	'0'	'0'
'1'	'0'	'1'	'1'
'1'	'1'	'0'	'0'
'1'	'1'	'1'	'1'

1.a) (0.25 puntos) Obtenga la función lógica `F` a partir de la tabla de verdad. Escriba en VHDL la **entity** del circuito que implemente la función lógica. Es decir, que tenga tres entradas `x`, `y` y `z`, y una salida `F`.

1.b) (0.75 puntos) Escriba en VHDL la **architecture** que describa el *comportamiento* del circuito.

1.c) (0.25 punto) Dibuje el diagrama de un circuito que implemente esta función lógica 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) (0.75 puntos) 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 para comprobar los circuitos diseñados en los Apartados 1.b y 1.d.

Solución al Ejercicio 1

La **entity** del circuito que implementa la función lógica 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 funcF is
   port ( F: out std_logic;
               x, y, z  : in std_logic );
end entity funcF;
-------------------------------------
```

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

```
-------------------------------------
--F = x'y'+xz
-------------------------------------
architecture Comp of funcF is
begin
  F <= (not x    and not y)
          or (x    and z);
end architecture Comp;
-------------------------------------
```

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

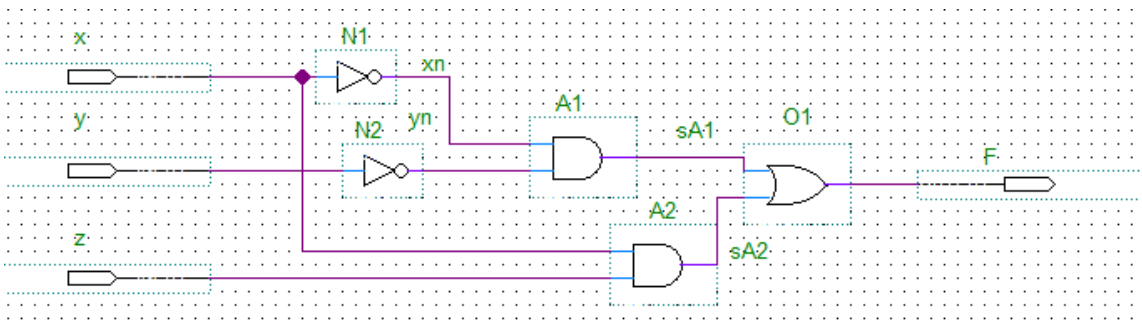

Figura 1.1: Solución al Apartado 1.c: diagrama al nivel de puertas lógicas.

La Figura 1.1 muestra el diagrama del circuito implementado empleando dos puertas AND de dos entradas, dos puertas NOT y una puerta OR 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.

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

```
  ------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito_Estruc of funcFG is
   signal xn, yn: std_logic;
   signal sA1, sA2: 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 (xn, x);
   N2 : component not1 port map (yn, y);
   A1 : component and2 port map (sA1, xn, yn);
   A2 : component and2 port map (sA2, x, z);
   O1 : component or2 port map (F, sA1, sA2);
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 aplicado al diseño del Apartado 1.b. y 1.d se muestran, respectivamente, en las Figuras 1.2 y 1.3.

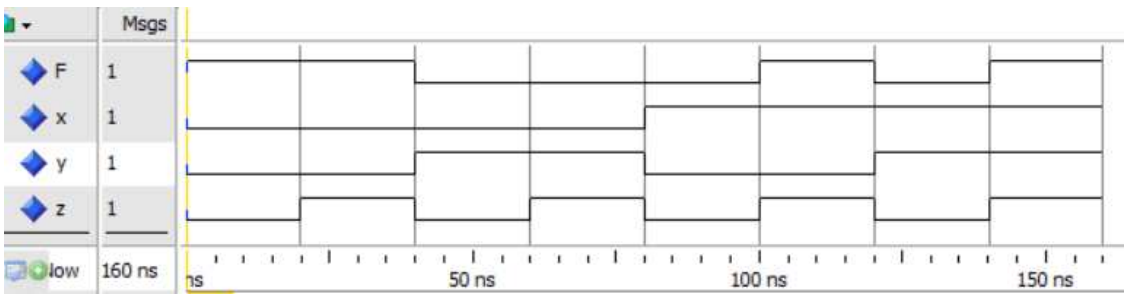

Figura 1.2: Cronograma del banco de pruebas aplicado al diseño del Apartado 1.b.

```
----------------------------------------------------
-- Banco de pruebas del circuito Ejercicio 1
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

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

architecture bp_funcF of bp_funcF is
    signal F: std_logic;
    signal x, y, z    : std_logic;

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

begin
    UUT : component funcF port map
          (F, 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_funcF;
----------------------------------------------------
```

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

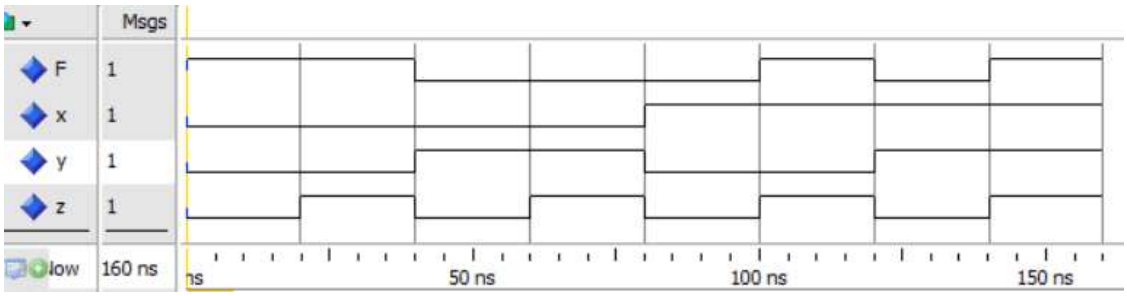

Figura 1.3: Cronograma del banco de pruebas aplicado al diseño del Apartado 1.d.

EJERCICIO 2 (7 PUNTOS)

Se pretende diseñar un circuito combinacional para controlar el disparo de las alarmas de una casa. La casa tiene 3 alarmas: de incendios, antirrobo y de detección de fugas de agua. El circuito tiene 7 señales de entrada: 5 señales de entrada de un bit procedentes de sensores y 2 entradas cuya función es deshabilitar las alarmas. El circuito tiene 3 señales de salida de un bit cuyo objetivo es disparar las 3 alarmas.

A continuación se describe el significado de las 7 señales de entrada del circuito:

- Señal `smoke`: tiene valor '1' sólo si se detecta humo.
- Señal `front_door`: tiene valor '1' sólo si se detecta que la puerta delantera está abierta.
- Señal `back_door`: tiene valor '1' sólo si se detecta que la puerta trasera está abierta.
- Señal `side_door`: tiene valor '1' sólo si se detecta que la puerta lateral está abierta.
- Señal `water_detect`: tiene valor '1' sólo si se detecta que existe una fuga de agua.
- Señal `alarm_disable`: si tiene valor '1', deshabilita únicamente el sistema de alarma antirrobo.
- Señal `main_diable`: si tiene valor '1', deshabilita las 3 alarmas. Es decir, deshabilita la alarma antirrobo, de incendios y de fugas de agua.

A continuación se describe el significado de las 3 las señales de salida del circuito:

- Señal `fire_alarm`: se pone a '1' sólo si se detecta humo y está habilitada la alarma de incendios. En caso contrario, tiene valor '0'.
- Señal `burg_alarm`: se pone a '1' sólo si se detecta que alguna de las 3 puertas está abierta (delantera, trasera o lateral) y además está habilitada la alarma antirrobo. En caso contrario, tiene valor '0'.
- Señal `water_alarm`: se pone a '1' sólo si se detecta que existe una fuga de agua y está habilitada la alarma de fugas de agua. En caso contrario, tiene valor '0'.

2.a) (0.5 puntos) Escriba en VHDL la **entity** del circuito de control. Todas las señales de entrada y de salida del circuito han de ser del tipo `std_logic`.

2.b) (2 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito empleando un bloque **process** y sentencias **if**.

2.c) (2 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito empleando únicamente sentencias de asignación concurrente.

2.d) (2 puntos) Dibuje el diagrama del circuito 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 una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando las puertas lógicas que ha diseñado anteriormente.

2.e) (0.5 puntos) 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 2.b, 2.c y 2.d. Compruebe mediante inspección visual que los tres diseños funcionan correctamente. Incluya en la memoria los tres cronogramas obtenidos al realizar la simulación del banco de pruebas para comprobar los circuitos diseñados en los Apartados 2.b, 2.c y 2.d.

Solución al Ejercicio 2

El código VHDL de la **entity** del circuito de control se muestra en Código VHDL 1.8. La **architecture** que describe el comportamiento del circuito empleando un bloque **process** y sentencias **if** se muestra en Código VHDL 1.9. La **architecture** que describe el comportamiento del circuito empleando únicamente sentencias de asignación concurrentes se muestra en Código VHDL 1.10.

La Figura 1.4 muestra el diagrama circuital del circuito de control. Este circuito consta de dos puertas NOT, una puerta OR de 3 entradas y 2 puertas AND de dos entradas. La **architecture** de las puertas NOT, AND y OR se muestran respectivamente en Código VHDL 1.3, 1.4 y 1.11.

La **architecture** que describe la *estructura* del circuito se muestra en Código VHDL 1.12.

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

entity control is
   port ( fire_alarm, burg_alarm, water_alarm: out std_logic;
          smoke, front_door, back_door, side_door,
          alarm_disable, main_disable, water_detect : in std_logic );
end entity control;
----------------------------------
```

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

```
----------------------------------
--Sistema de control con IF
----------------------------------
architecture Comp of control is
begin
 process(smoke, front_door, back_door, side_door,
          alarm_disable, main_disable, water_detect)
 begin
    if((smoke ='1')and (main_disable = '0')) then
      fire_alarm <= '1';
    else
       fire_alarm <= '0';
    end if;
    if (((front_door='1') or (back_door='1')
        or (side_door='1')) and ((alarm_disable='0')
        and (main_disable='0'))) then
       burg_alarm <= '1';
    else
       burg_alarm <= '0';
    end if;
    if ((water_detect = '1') and (main_disable = '0')) then
      water_alarm <='1';
    else
      water_alarm <= '0';
    end if;
end process;
end architecture Comp;
----------------------------------
```

Código VHDL 1.9: Solución al Apartado 2.b: **architecture** del circuito de control empleando un bloque **process**.

```
-------------------------------------
--Sistema de control describiendo
--la estructura con puertas logicas
-------------------------------------
architecture Concurrente of control is
begin
fire_alarm <= (smoke and (not(main_disable)));
water_alarm <= water_detect and not(main_disable);
burg_alarm <= ( front_door or back_door or side_door) and
              (not (alarm_disable) and (not (main_disable)) );
end architecture Concurrente;
-------------------------------------
```

Código VHDL 1.10: Solución al Apartado 2.c: **architecture** del circuito de control empleando sentencias de asignación concurrente.

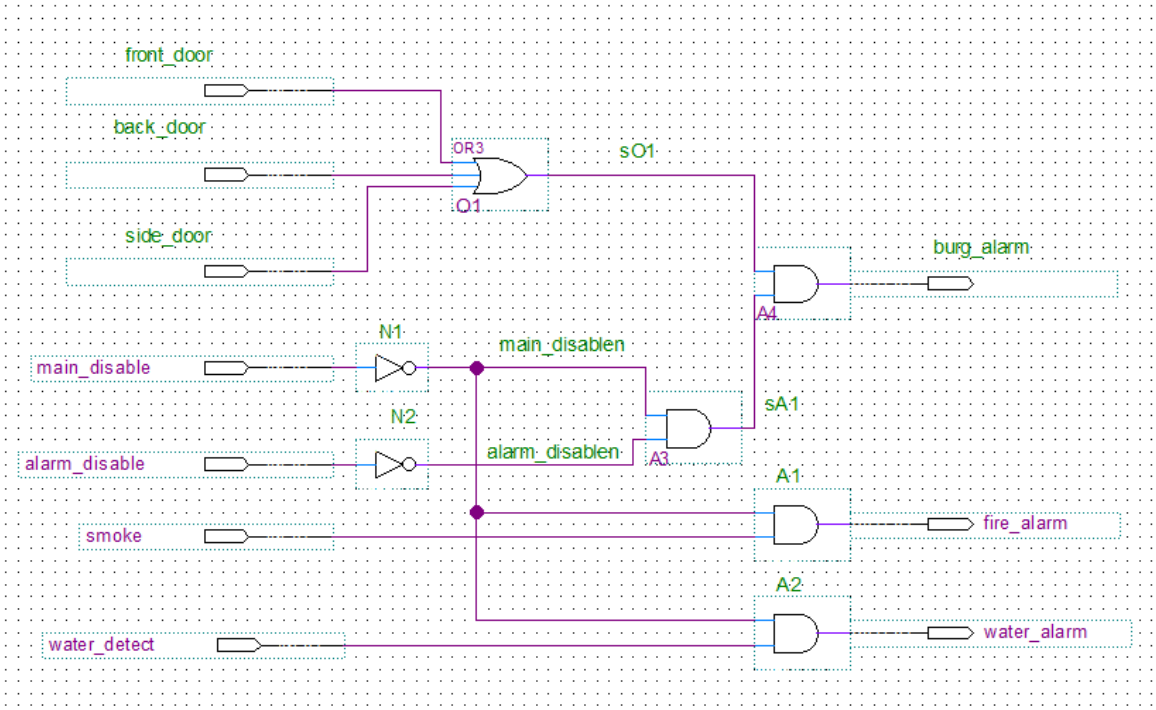

Figura 1.4: Diagrama circuital solución del Apartado 2.d.

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

entity or3 is
   generic ( DELAY : time := 10 ns );
   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 ) after DELAY;
end architecture or3;
-------------------------------------
```

Código VHDL 1.11: Puerta lógica OR de 3 entradas.

```
-------------------------------------
--Sistema de control describiendo
--su estructura
-------------------------------------
architecture Estructural of control is
   signal main_disablen, alarm_disablen: std_logic;
   signal sO1, sA1: 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 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 (main_disablen, main_disable);
   N2 : component not1 port map (alarm_disablen, alarm_disable);
   A1 : component and2 port map (fire_alarm, smoke, main_disablen);
   A2 :   component   and2 port   map   (water_alarm,   water_detect,
main_disablen);
   O1 : component or3 port map (sO1, front_door, back_door, side_door);
   A3 : component and2 port map (sA1, alarm_disablen, main_disablen);
   A4 : component and2 port map (burg_alarm, sO1, sA1);
end architecture Estructural;
-------------------------------------
```

Código VHDL 1.12: Solución al Apartado 2.d: Diseño estructural del circuito de control.

El banco de pruebas del circuito se muestra en Código VHDL 1.13. El cronograma obtenido tras simular el banco de pruebas con cada uno de los 3 diseños del circuito de control realizado se muestran en 1.5–1.7.

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

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

architecture bp_control of bp_control is
    signal fire_alarm, burg_alarm, water_alarm: std_logic;
    signal smoke, front_door, back_door, side_door  : std_logic;
    signal alarm_disable, main_disable, water_detect  : std_logic;

    component control is
     port ( fire_alarm, burg_alarm, water_alarm: out std_logic;
           smoke, front_door, back_door, side_door,
           alarm_disable, main_disable, water_detect : in std_logic );
    end component control;


begin
    UUT : component control port map
          (fire_alarm, burg_alarm, water_alarm, smoke, front_door,
back_door,
           side_door, alarm_disable, main_disable, water_detect);

vec_test : process is
      variable valor  : unsigned (6 downto 0);
    begin
      -- Generar todos los posibles valores de entrada
      for i in 0 to 127 loop
        valor := to_unsigned(i,7);
        alarm_disable <= std_logic(valor(6));
        main_disable <= std_logic(valor(5));
        water_detect <= std_logic(valor(4));
        smoke <= std_logic(valor(3));
        front_door <= std_logic(valor(2));
        back_door <= std_logic(valor(1));
        side_door <= std_logic(valor(0));
        wait for DELAY;
      end loop;
      wait;     -- Final de la simulación
   end process vec_test;
end architecture bp_control;
-------------------------------------------------
```

Código VHDL 1.13: Solución al Apartado 2.e: Diseño del banco de pruebas.

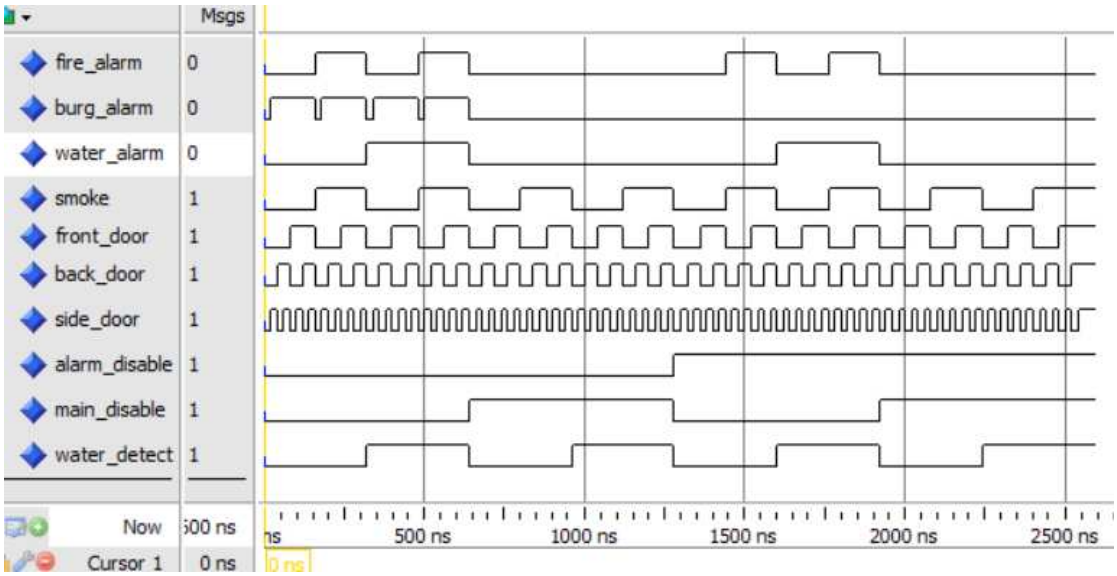

Figura 1.5: Cronograma obtenido al testear el circuito de control diseñado empleando un bloque **process**.

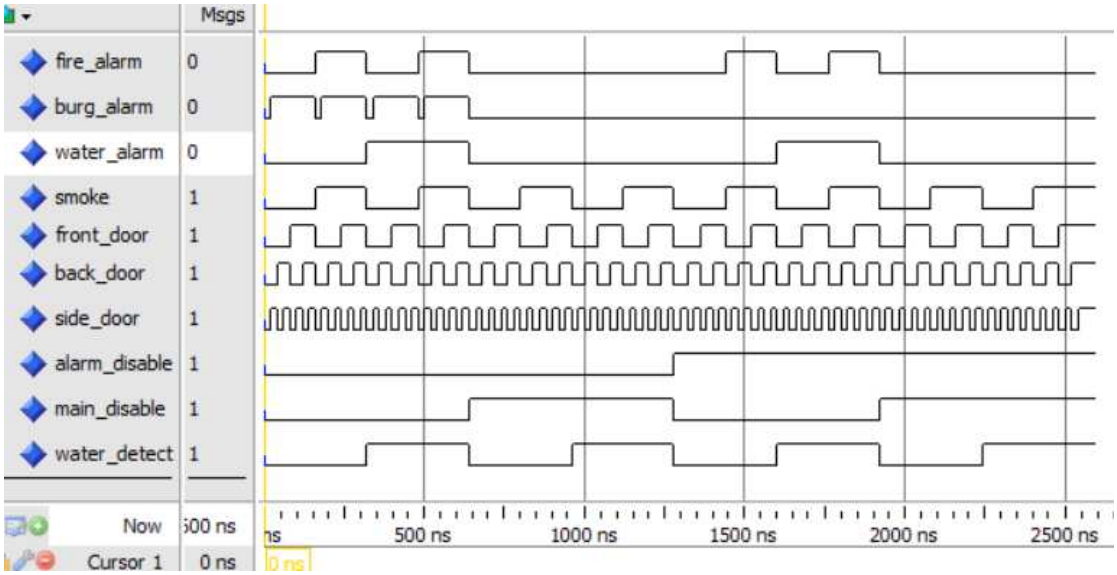

Figura 1.6: Cronograma obtenido al testear el circuito de control diseñado empleando sentencias de asignación concurrentes.

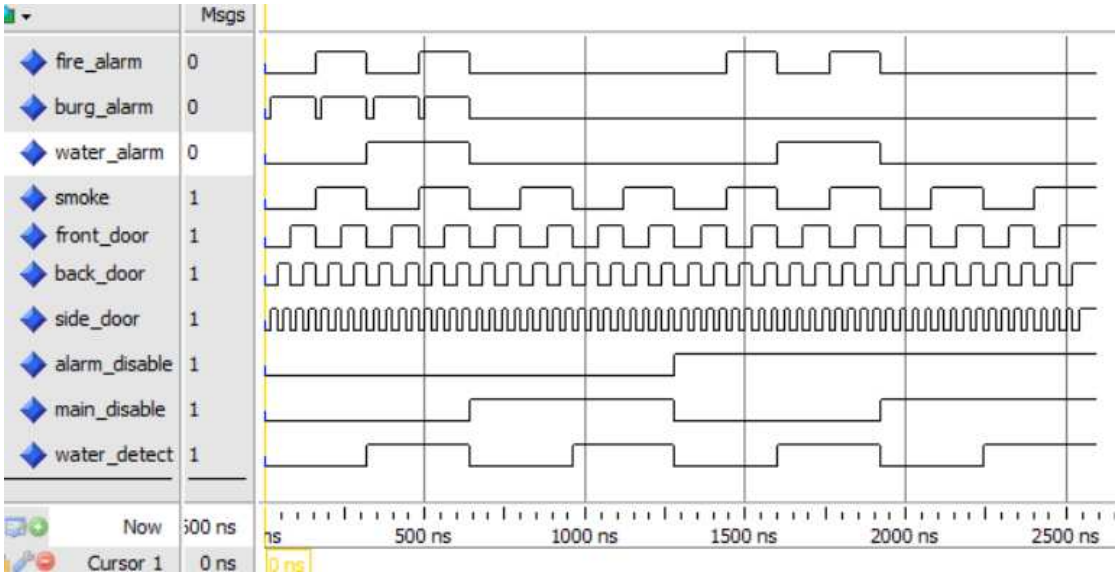

Figura 1.7: Cronograma obtenido al testear el circuito de control descrito empleando puertas lógicas.